

**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**



**Grado en Ingeniería Informática**

## **TRABAJO FIN DE GRADO**

**DESARROLLO DE UNA APLICACIÓN WEB PARA VENTA  
DE PRODUCTOS DE PEQUEÑOS COMERCIOS**

**David García García**  
**Tutor: Miguel Ángel Mora Rincón**

**Abril 2020**



# **DESARROLLO DE UNA APLICACIÓN WEB PARA VENTA DE PRODUCTOS DE PEQUEÑOS COMERCIOS**

**AUTOR: David García García**  
**TUTOR: Miguel Ángel Mora Rincón**

**Dpto. de Ingeniería Informática**  
**Escuela Politécnica Superior**  
**Universidad Autónoma de Madrid**  
**Abril 2020**



# Resumen (castellano)

En la actualidad numerosos trabajadores autónomos y pequeñas empresas encuentran difícil promocionar sus comercios, así como aumentar los clientes y las ventas de productos. Mientras, las grandes compañías con la ayuda de la publicidad y la venta online han crecido exponencialmente en los últimos años, haciendo más difícil la situación de estos pequeños comercios.

Un grupo de pequeños comercios de un barrio de San Sebastián de los Reyes han propuesto la creación de una web que sea un punto de acceso a sus negocios en internet con el fin de aumentar su visibilidad y ventas para superar esta situación tan compleja por la que están pasando.

En cuanto al apartado técnico, la aplicación va a ser una aplicación PWA (*Progressive Web Application*) desarrollada mediante una de las tecnologías más punteras en el desarrollo de aplicaciones reactivas basadas en Javascript: Nuxt.js, junto con la computación en la nube mediante Firebase y el manejo de bases de datos NoSQL. El autor de este proyecto se pondrá en la situación de consultor que se reunirá con el cliente y analizará las necesidades para realizar un análisis y diseño conciso de la aplicación con el fin de mostrar la mejor solución posible al cliente. También hay que destacar el trabajo de aprendizaje por parte del autor de las tecnologías empleadas para el desarrollo de este proyecto.

Este documento recoge las fases de vida de un proyecto software, junto con una introducción que mostrará los objetos y motivación del proyecto, y una descripción de todas las posibles tecnologías que pueden utilizarse para la resolución del problema en el estado del arte.

El análisis está compuesto por la descripción del sistema, los requisitos obtenidos de las reuniones con el cliente y los casos de uso de la aplicación. En cuanto al diseño, este está focalizado en una pequeña parte de la aplicación con el fin de obtener un MVP (*Minimum Viable Product*), constará de las maquetas y el modelo de datos de la aplicación, así como la explicación de las tecnologías a utilizar en el desarrollo de la aplicación.

Por otro lado, la planificación realizada del proyecto será de cinco iteraciones de tres semanas de duración cada una, en la que se implementará la distinta funcionalidad acordada. A continuación, se relatará la inicialización del proyecto, su desarrollo y su posterior despliegue. Tras esto, se mostrarán las pruebas y los resultados obtenidos, finalizando con las conclusiones y trabajos futuros del proyecto para acabar de implementar la aplicación completa.



# Abstract (English)

Nowadays, many freelance workers and small companies find some difficulties when advertising their businesses and incrementing their clients and sales. Otherwise, big companies and online marketplaces have been growing exponentially in the last years, creating a tough situation for small businesses.

A group of small companies from a neighborhood of San Sebastián de los Reyes, have suggested the creation of a web that acts as an access point to their businesses. In addition, the web will be very helpful to advertise their shops and increment the sales. The group expects the platform to be a useful tool to overtake the current situation.

About the technical section, the application will be a PWA (*Progressive Web Application*) application and developed using some of the most recent technologies on the reactive application web development based in Javascript: Nuxt.js, as well as Firebase cloud computing Firebase, and NoSQL databases. On the one hand, the author of the project will act as a consultant, having meetings with clients and analysing their problems with the objective of analysing and designing an application that shows the best solution to the problem. On the other hand, the author will learn the technologies that will be used in order to develop the web.

This document includes the life stages of the software project joined an introduction shows the main objective and the motivation of the project and novel technologies are described in the State of the art section.

The analysis is made up of description of system, the requirements obtained in meetings with client and use cases of the application. About the design, it is focus on small part of application was designed in order to get a MVP (*Minimum Viable Product*) to develop this project. It contain the models made to develop the project, the database design and the explanation of the technologies to be used in the development of the application.

On the other hand, the project planning has five iterations of three week each where the agreed functionality will be performed. Then, come the initialization of the project, the develop stage and the project deployment. The document ends with the tests and results over develop and the conclusions and future works.





## Palabras clave (castellano)

Online, Javascript, Computación, Nube, Firebase, NoSQL, MVP, framework, Nuxt.js, maqueta, aplicación, web, multiplataforma, responsive, Vue.js, Nuxt.js, Vuex.js, componente, página, frontend, backend, cliente, servidor, iteración, estado, reactivo, HTML, CSS, Vuetify.js, MVVM, enrutamiento, demo, reglas, Host, PWA.

## Keywords (inglés)

Online, Javascript, computing, cloud, Firebase, NoSQL, MVP, framework, Nuxt.js, model, application, web, multiplatform, responsive, Vue.js, Nuxt.js, Vuex.js, component, page, frontend, backend, client, server, iteration, state, reactive, HTML, CSS, Vuetify.js, MVVM, routing, demo, rules, Host, PWA.



## ***Agradecimientos***

En primer lugar, quiero agradecer a Miguel Ángel Mora, mi tutor tanto del grado como de este proyecto, por permitirme realizar este proyecto, ya que es una idea que tenía durante varios años y al fin la he podido realizar, así como por guiarme durante todos estos años de grado y, en especial estos últimos meses.

Agradecer también a mis padres, a mi hermana, a mis abuelos y tíos por el apoyo proporcionado durante todos estos años y poniendo todos sus recursos disponibles en que alcanzase este objetivo que tan cerca tengo de conseguir.

Tampoco me quiero olvidar de mis compañeros y amigos conseguidos en estos años, hemos tenido muy buenos momentos y anécdotas, pero también hemos tenido momentos difíciles, pero siempre hemos puesto lo mejor de cada uno para salir. Gracias por el apoyo recibido por vosotros.

Para finalizar, quiero agradecer a la Universidad Autónoma de Madrid, en especial a la Escuela Politécnica Superior por haberme dado la posibilidad hace 6 años de cursar este grado en sus instalaciones, así como agradecer a todos los profesores que me han guiado durante esta etapa por todas las asignaturas ayudando a mi desarrollo como informático y como persona.



# INDICE DE CONTENIDOS

1	Introducción .....	1
1.1	Motivación.....	1
1.2	Objetivos.....	1
1.3	Organización de la memoria.....	2
2	Estado del arte .....	3
2.1	Plataformas actuales .....	3
2.1.1	Amazon Marketplace.....	3
2.1.2	1&1 Ionos.....	3
2.2	Tipos de aplicaciones.....	3
2.2.1	Aplicación web.....	3
2.2.2	Aplicación móvil .....	4
2.2.3	Aplicación híbrida .....	4
2.3	Framework de frontend.....	5
2.3.1	Angular.....	5
2.3.2	React.js .....	5
2.3.3	Vue.js + Nuxt.js.....	5
2.4	Arquitectura backend.....	6
2.4.1	Persistencia.....	6
2.4.2	Hosting .....	6
2.4.3	Backendless.....	7
3	Análisis.....	9
3.1	Reunión con el cliente .....	9
3.2	Descripción de la aplicación.....	9
3.3	Análisis de requisitos.....	10
3.3.1	Requisitos funcionales.....	10
3.3.2	Requisitos no funcionales.....	14
3.4	Casos de usos.....	14
4	Diseño .....	17
4.1	Tecnologías y lenguajes empleados.....	17
4.2	Maquetas.....	17
4.3	Arquitectura de la aplicación .....	18
4.3.1	Frontend .....	18
4.3.2	Backend .....	19
4.3.3	Persistencia.....	19
5	Planificación.....	23
5.1	Iteración 0: Análisis y diseño de la aplicación .....	23
5.2	Iteración 1: Superusuario .....	23
5.3	Iteración 2: Gestión de usuarios .....	23
5.4	Iteración 3: Gestión de comercios .....	23
5.5	Gestión de artículos .....	24
5.6	Iteración 5: Carrito de la compra y finalización pedidos.....	24
6	Desarrollo.....	25
6.1	Entorno de desarrollo.....	25
6.2	Inicialización de la aplicación .....	25
6.3	Frontend.....	26
6.3.1	Components.....	26
6.3.2	Pages.....	27
6.3.3	Store .....	28
6.4	Backend .....	29
6.4.1	Hosting .....	29
6.4.2	Reglas de seguridad.....	30
6.4.3	Firebase functions.....	31

6.5 Despliegue .....	31
7 Integración, pruebas y resultados .....	33
7.1 Integración .....	33
7.2 Pruebas.....	33
7.3 Demos continuas.....	34
7.4 Resultados.....	34
8 Conclusiones y trabajo futuro .....	37
8.1 Conclusiones.....	37
8.2 Trabajo futuro .....	38
Referencias .....	39
Glosario .....	41
Anexos.....	I
A Casos de uso .....	I
B Maquetas .....	IX
C Planificación .....	XIII
D Consola Firebase.....	XV
E Componentes de la aplicación.....	XXI
F Componente Vue.js .....	XXIII
G Páginas de la aplicación.....	XXV
H Resultado interfaz gráfica de la aplicación .....	XXVII

# INDICE DE FIGURAS

FIGURA 1. DIAGRAMA DE CASOS DE USO USUARIO SIN ROL .....	14
FIGURA 2. DIAGRAMA DE CASOS DE USO USUARIO CON ROL SUPERUSUARIO.....	15
FIGURA 3. DIAGRAMA DE CASOS DE USO USUARIO CON ROL ADMINISTRADOR. ....	15
FIGURA 4. DIAGRAMA DE CASOS DE USO USUARIO CON ROL EMPLEADO.....	16
FIGURA 5. DIAGRAMA DE CASOS DE USO USUARIO CON ROL USUARIO. ....	16
FIGURA 6. MAQUETA DE LA VISTA INICIAL DE LA APLICACIÓN. ....	17
FIGURA 7. INTEGRACIÓN DE MVVM CON FLUX.JS.....	18
FIGURA 8. EJEMPLO JSON DOCUMENTO USER. ....	20
FIGURA 9. EJEMPLO JSON DOCUMENTO SHOP. ....	20
FIGURA 10. EJEMPLO JSON DOCUMENTO ORDER .....	20
FIGURA 11. EJEMPLO JSON DOCUMENTO ARTICLE.....	21
FIGURA 12. ESQUEMA MODELO DE DATOS.....	21
FIGURA 13. COMPONENTS DE LA APLICACIÓN.....	26
FIGURA 14. EJEMPLO DE ESCUCHA DE COLECCIONES. ....	26
FIGURA 15. EJEMPLO DE MAPPEO DE LAS FUNCIONES DEL STORE DENTRO DE UN COMPONENT. ....	27
FIGURA 16. EJEMPO DE DECLARACIÓN DE COMPONENTE CON PROPIEDAD. ....	27
FIGURA 17. EJEMPLO DE DECLARACIÓN DE PROPS.....	27
FIGURA 18. PÁGINAS DE LA APLICACIÓN. ....	27
FIGURA 19. EJEMPLO DE PÁGINAS ANIDADAS. ....	28
FIGURA 20. STORE EXISTENTES EN EL PROYECTO.....	28
FIGURA 21. EJEMPLO DE FUNCIONES ACTIONS. ....	28
FIGURA 22. FICHEROS JAVASCRIPT PARA LA CONEXIÓN CON FIREBASE. ....	29
FIGURA 23. EJEMPLO DE FUNCIONES DE API.JS .....	29
FIGURA 24. REGLAS DE SEGURIDAD IMPLEMENTADAS. ....	30
FIGURA 25. PROYECTO DE FIREBASE FUNCTIONS.....	31

FIGURA 26. EJEMPLO DE FUNCIÓN DE FIREBASE. ....	31
FIGURA 27. RESULTADO DEL DESPLIEGUE. ....	32
FIGURA 28. MAQUETA DE LA VISTA ARTÍCULOS. ....	IX
FIGURA 29. MAQUETA DEL INVENTARIO ARTÍCULOS ASOCIADOS AL COMERCIO. ....	IX
FIGURA 30. MAQUETA DEL LISTADO EMPLEADOS COMERCIO. ....	X
FIGURA 31. MAQUETA DE LA LISTA PEDIDOS ASOCIADOS AL COMERCIO. ....	X
FIGURA 32. MAQUETA DE LA VISTA PRINCIPAL CON MENÚ DESPLEGABLE. ....	X
FIGURA 33. MAQUETA DE LA VISTA PRINCIPAL CON MENÚ DE USUARIO. ....	XI
FIGURA 34. MAQUETA DE LA VISTA PRINCIPAL USUARIO LOGADO. ....	XI
FIGURA 35. MAQUETA DE LA VISTA PRINCIPAL COMERCIO ....	XI
FIGURA 36. CALENDARIO DE LA PLANIFICACIÓN. ....	XIII
FIGURA 37. PÁGINA PRINCIPAL FIREBASE. ....	XV
FIGURA 38. LISTA DE PROYECTOS FIREBASE. ....	XV
FIGURA 39. VENTANA PRINCIPAL PROYECTO FIREBASE. ....	XVI
FIGURA 40. PÁGINA DE USUARIOS FIREBASE. ....	XVI
FIGURA 41. PÁGINA DE TIPOS DE INICIO DE SESIÓN FIREBASE AUTHENTICATION. ....	XVII
FIGURA 42. VISTA CLOUD FIRESTORE EN CONSOLA FIREBASE. ....	XVII
FIGURA 43. VISTA REGLAS SEGURIDAD FIREBASE. ....	XVIII
FIGURA 44. PANEL DE CONTROL HOSTING CONSOLA FIREBASE. ....	XVIII
FIGURA 45. PANEL FUNCTIONS EN CONSOLA. ....	XIX
FIGURA 46. CICLO DE VIDA COMPONENTE VUE.JS. ....	XXIII
FIGURA 47. UI DE LA PÁGINA PRINCIPAL. ....	XXVII
FIGURA 48. UI DE LA PÁGINA PRINCIPAL CON EL CARRITO DESPLEGADO. ....	XXVII
FIGURA 49. UI DEL INICIO DE SESIÓN. ....	XXVIII
FIGURA 50. UI DEL REGISTRO. ....	XXVIII
FIGURA 51. UI DE LA PÁGINA DE AYUDA. ....	XXVIII



FIGURA 52. UI DE LA VISTA DE UN COMERCIO.....	XXIX
FIGURA 53. UI DE LA VISTA DE UN ARTÍCULO.....	XXIX
FIGURA 54. UI DEL FORMULARIO DE REGISTRO DE COMERCIO.....	XXIX
FIGURA 55. UI DEL PERFIL DE USUARIO. ....	XXX
FIGURA 56. UI DEL FORMULARIO DE MODIFICAR UN USUARIO.....	XXX
FIGURA 57. UI DEL USUARIO DE LA LISTA DE PEDIDOS ASOCIADOS. ....	XXX
FIGURA 58. UI DEL ADMINISTRADOR DE LA LISTA DE ARTÍCULOS ASOCIADOS.....	XXXI
FIGURA 59. UI DEL FORMULARIO PARA AÑADIR UN NUEVO ARTÍCULO.....	XXXI
FIGURA 60. UI DEL FORMULARIO PARA MODIFICAR UN ARTÍCULO. ....	XXXI
FIGURA 61. UI DEL ADMINISTRADOR DE LA LISTA DE PEDIDOS ASOCIADA. ....	XXXII
FIGURA 62. UI DEL PERFIL DE UN COMERCIO.....	XXXII
FIGURA 63. UI DEL FORMULARIO PARA MODIFICAR UN COMERCIO.....	XXXII
FIGURA 64. UI DEL SUPERADMINISTRADOR DE LA LISTA DE USUARIOS. ....	XXXIII
FIGURA 65. UI DEL SUPERADMINISTRADOR DE LISTA DE ARTÍCULOS. ....	XXXIII
FIGURA 66. UI SUPERADMINISTRADOR DE LA LISTA DE PEDIDOS.....	XXXIII
FIGURA 67. UI SUPERADMINISTRADOR DE LA LISTA DE COMERCIOS.....	XXXIV



## INDICE DE TABLAS

TABLA 1. CASO DE USO CU-01 .....	I
TABLA 2. CASO DE USO CU-02 .....	II
TABLA 3. CASO DE USO CU-03 .....	III
TABLA 4. CASO DE USO CU-04 .....	IV
TABLA 5. CASO DE USO CU-05 .....	V
TABLA 6. CASO DE USO CU-06 .....	VI
TABLA 7. CASO DE USO CU-07 .....	VII
TABLA 8. CASO DE USO CU-08 .....	VIII
TABLA 9. LISTA DE LOS COMPONENTES DE LA APLICACIÓN. ....	XXI
TABLA 10. LISTA DE PÁGINAS DE LA APLICACIÓN. ....	XXV
TABLA 11. LISTA DE RUTAS DE LA APLICACIÓN.....	XXVI



# 1 Introducción

---

## 1.1 Motivación

Esta memoria de TFG trata sobre el análisis, diseño y desarrollo de una aplicación web para ayudar a las pequeñas empresas y autónomos de un barrio de *San Sebastián de los Reyes (Madrid)* para que puedan darse a conocer fuera del barrio y, además, puedan publicar sus productos y realizar ventas online, con el fin de mejorar las ventas de sus comercios.

En particular, la madre del autor de este Trabajo Fin de Grado (TFG) regenta una mercería (*El Desván de Yoly*) ha visto como han disminuido las ventas al público en este último año y ha solicitado este, la creación de una aplicación web con el fin de darse a conocer fuera del barrio y poder vender sus productos vía online. Además, exponiendo a los comercios vecinos la idea de crear esta plataforma, se han sumado a la petición, ya que han visto una gran oportunidad de poder tener un nuevo escaparate para el público y así, relanzar las ventas perdidas.

En la actualidad, existen varias plataformas para la venta de productos en España, como *Amazon Marketplace*, también existen otras plataformas para crear aplicaciones web personalizadas para empresas, como *1&1 Ionos*, incluso se puede contratar a una empresa o desarrollador web para que realice dicho trabajo. Pero estas plataformas o desarrollos de aplicaciones tienen un alto coste para estos pequeños comercios que no pueden permitirse. Por ello, este grupo de personas han solicitado al autor de este TFG la creación de una plataforma propia donde se incluyan todos los comercios de la zona con el objetivo de aumentar las ventas y los ingresos del negocio.

## 1.2 Objetivos

El principal objetivo de este TFG es el análisis, diseño e implementación de una aplicación web multiplataforma para incrementar el número de ingresos de las pequeñas empresas y autónomos de un barrio San Sebastián de los Reyes. Además, se ha propuesto como objetivo realizar este proyecto de la forma más parecida a un ámbito profesional. Para ello, los gerentes y autónomos de los comercios actuarán como cliente y el autor de este TFG como consultor y desarrollador del proyecto. Gracias a esto, el alumno va a experimentar el trato con un cliente final y las relaciones con este en entornos más abiertos a los encontrados a lo largo del grado, además de tener que cumplir los plazos acordados con el cliente y entregar un producto de calidad.

Es necesario que este proyecto se realice en iteraciones donde se implemente funcionalidad cerrada y modular con el fin de que, al finalizar una iteración, se realicen demos con el cliente y que este muestre sus opiniones de lo presentado para, en caso de que fuera necesario realizar los cambios pertinentes lo antes posible.

Cabe destacar que, el análisis realizado será de todo el proyecto, donde se llevarán a cabo todas las reuniones necesarias con el cliente con el fin de obtener todos los datos necesarios para, posteriormente plasmarlos mediante un análisis de requisitos y diagramas de casos de uso. Pero, debido a la envergadura de este proyecto, el diseño e implementación se va a centrar en realizar un *Minimum Viable Product* MVP de la aplicación que se centrarán en la gestión de los usuarios, comercios y artículos, así como la posibilidad de realizar pedidos mostrando la funcionalidad básica de esta aplicación. En posteriores iteraciones que se realizarán fuera de este TFG, la funcionalidad realizada, aumentará incluyendo la conexión con pasarelas de pago, inclusión de comentarios y valoraciones de producto y la inclusión de métodos de envío, así como el resto de funcionalidad analizada y no incluida en la planificación.

## 1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción.** En este capítulo se trata la motivación y los objetivos del proyecto con el fin de poner en contexto al leyente sobre este documento.
- **Estado del arte.** En el estado del arte se realiza una descripción del estudio realizado sobre las distintas tecnologías existentes en la actualidad para resolver el problema planteado.
- **Análisis.** En este apartado, uno de los más importantes de este documento, se realizará una descripción detallada sobre la funcionalidad del proyecto completo, incluyendo el análisis de requisitos y los distintos diagramas de casos de uso realizados tras obtener los datos de distintas reuniones con el cliente.
- **Diseño.** En el apartado de Diseño, se explica las tecnologías utilizadas para la implementación del proyecto y definidas en el estado del arte. Además, se incluirán distintas maquetas realizadas y mostradas al cliente para la interfaz gráfica y la descripción del modelo de datos de la aplicación. Junto con el punto anterior, es otro de los apartados más relevantes de este documento.
- **Planificación.** En este punto se tratará la planificación realizada previa al desarrollo de este proyecto. En él se encontrará una definición de las distintas iteraciones de las que consta el proyecto completo y su estimación en tiempo.
- **Desarrollo.** En este capítulo se plasmará un resumen del desarrollo de las iteraciones realizadas en este TFG. Se mostrará desde la creación del proyecto y los elementos más relevantes que lo componen, así como distinta funcionalidad relevante del *frontend* y *backend* y el despliegue de del proyecto. Junto con los capítulos de Análisis y Desarrollo, son los capítulos con más relevancia de esta memoria.
- **Integración, pruebas y resultados.** Este apartado se centrará en la integración de la aplicación con agentes externos, las pruebas realizadas durante el desarrollo, incluyendo las distintas demos realizadas, y los resultados de estas.
- **Conclusiones y trabajos futuros.** Este último punto recoge las conclusiones obtenidas después de realizar el proyecto y los distintos trabajos que quedan por realizar hasta la finalización de la aplicación final.
- **Referencias.** Bibliografía consultada para la realización del proyecto.
- **Glosario.** Siglas utilizadas en este documento.
- **Anexos:**
  - **Casos de uso.** Ocho casos de uso realizados en la fase de análisis.
  - **Maquetas.** Resto de maquetas realizadas en la fase de diseño.
  - **Planificación.** Explicación de la planificación: obtención de jornadas, división de tareas y calendario.
  - **Consola Firebase.** Explicación del funcionamiento de la consola Firebase.
  - **Componentes de la aplicación.** Listado de los componentes desarrollados en el proyecto.
  - **Componente Vue.js.** Explicación de la lógica de un componente Vue.js y su ciclo de vida.
  - **Páginas de la aplicación.** Listado de las páginas y rutas de la aplicación.
  - **Resultado interfaz gráfica de la aplicación.** Conjunto de imágenes resultantes del desarrollo de la aplicación.

## 2 Estado del arte

---

En este capítulo vamos a mostrar una serie de alternativas tecnológicas para la resolución del problema planteado en este proyecto.

### 2.1 Plataformas actuales

En la actualidad ya existe en el mercado una serie de aplicaciones para que cualquier comercio pueda darse a conocer o vender sus productos. En este punto, se va a visualizar algunos ejemplos de estas plataformas y visualizar las principales diferencias de estas con la plataforma que se desea implementar en este TFG.

#### 2.1.1 Amazon Marketplace

Amazon es una compañía estadounidense fundada a mediados de la década de los noventa que, entre todos sus productos, tiene una aplicación de comercio electrónico: *Amazon Marketplace*. Esto permite a los que desee, crear una cuenta y publicar sus productos para que quien lo desee pueda vender dichos productos. El coste de esta cuenta es de 39€/mes más IVA y proporciona un portal para la compraventa de todo tipo de productos, sin espacio propio de publicidad a los comercios. En cambio, la plataforma que se desea implementar consta de un espacio propio para cada comercio registrado. [1] [2] [3]

#### 2.1.2 1&1 Ionos

*1&1 Ionos* es una compañía alemana, anteriormente conocida como *1&1 MyWeb*, que proporciona un servicio de creación y hosting de aplicaciones web para cualquier usuario que se registre en su aplicación. Esta te permite personalizar una aplicación web y su posterior hosting, pero sin la posibilidad de incluir venta de productos en esta aplicación. El coste de una cuenta en *1&1 Ionos* es de aproximadamente 15€/mes (dependiendo de la tarifa seleccionada). [4]

Esta plataforma se centra en la creación de aplicaciones sin tener foco en comercios y compraventa de productos, aunque actualmente se está desarrollando un comercio online. Cabe destacar que esta web se publica independientemente, sin ser una plataforma común para una serie de comercios. En cambio, la aplicación de la que trata este TFG es una plataforma común para todos los comercios que lo deseen y donde se incluye, una plataforma de comercio online junto con un espacio individual para cada comercio. [5]

### 2.2 Tipos de aplicaciones

Existen distintos tipos de aplicaciones para desarrollar con una serie de características, así como sus propias ventajas e inconvenientes. En este punto, se va a realizar una breve descripción de estas aplicaciones y se van a mostrar estas ventajas e inconvenientes.

#### 2.2.1 Aplicación web

Las aplicaciones web son programas desarrollados en lenguajes soportados por un navegador web y ejecutados por estos o mediante una intranet. Este tipo de aplicaciones se caracterizan, generalmente, por tener independencia del Sistema Operativo y un cliente ligero que llama a servidores para obtener la información. Pero esto, como se verá más adelante, no es del todo cierto, ya que con la entrada de los *framework* basados en Javascript, el cliente está consiguiendo un peso considerable en las aplicaciones web. Además, estas aplicaciones facilitan el mantenimiento, debido a que las actualizaciones se aplican sobre el servidor y no es necesario la instalación de nuevo software en todas las computadoras que lo usen. [6] [28]

Las aplicaciones web, generalmente se dividen en cliente y servidor: en el lado cliente destaca el uso de HTML y CSS para la creación de la parte gráfica de la aplicación, y Javascript o PHP para la parte lógica existente en el cliente. Por otro lado, en el servidor son programas que pueden ser implementados en cualquier tipo de lenguaje: Java, Python, Javascript, C, C++... Además, por lo general, la persistencia de datos se realiza en el servidor mediante bases de datos relacionales o no relacionales, aunque el lado cliente también puede tener su propia persistencia. En cuanto a la comunicación entre el cliente y el servidor se suele utilizar el protocolo HTTP (HTTPS), que es el protocolo más utilizado en el internet de las cosas y permite la transferencia de datos en formato de texto. [7]

Existen varios tipos de aplicaciones web con características distintas que, dependiendo del uso que se quiera permitir en la aplicación es recomendable implementar uno u otro tipo:

- **Aplicaciones Cliente-Servidor.** Son aplicaciones web universales con una arquitectura donde el cliente es el encargado de mostrar las vistas y datos y el servidor tiene el peso computacional de la aplicación. En un flujo básico, el cliente solicita los datos al servidor mediante el protocolo HTTP, el servidor accede a la base de datos para obtenerlos, aplica una serie de reglas y devuelve una respuesta correcta con la información solicitada o una respuesta de error y, con la respuesta del servidor, el cliente muestra una vista u otra. [8]
- **Aplicaciones SPA.** Las aplicaciones SPA consisten en cargar al principio de la aplicación todo el código HTML, CSS y Javascript y los recursos se van cargando dinámicamente cuando el cliente los solicite. En este tipo de aplicaciones se busca que el usuario tenga una experiencia más fluida, ya que solo se puede cargar una página y lo que va cambiando es el contenido de esta. Para la obtención de los recursos se suele utilizar AJAX, que es un *framework* de llamadas asíncronas al servidor, por lo que el lado cliente deberá gestionar el estado de los recursos. [9]
- **Aplicaciones PWA.** Las aplicaciones PWA son aplicaciones SPA que permiten ser instaladas en los dispositivos como aplicaciones móviles o aplicaciones de escritorio. Esto provoca se pueda acceder a ellas sin necesidad de acceder a un navegador ofreciendo características similares a las de una aplicación nativa. [10] [28]

## 2.2.2 Aplicación móvil

Las aplicaciones móviles son aquellos programas desarrollados expresamente para poder ser ejecutados en dispositivos inteligentes como *smartphones*, *tablets* o *smartwatch*. En los últimos años, este tipo de aplicaciones ha tenido una alta aceptación en la sociedad debido a que este tipo de dispositivo son más utilizados que los ordenadores.

Por lo general, el desarrollo de este tipo de aplicaciones se asemeja a las aplicaciones web, pero debe tener en cuenta el software y hardware del dispositivo debido a que la aplicación será instalada en un dispositivo inteligente. Normalmente, este servidor tiene una arquitectura *backendless* y el cliente es el encargado de tener más peso dentro de la aplicación. Los lenguajes utilizados en el desarrollo de este tipo de aplicaciones son, por lo general C#, Javascript o, en los últimos años, un nuevo lenguaje denominado Kotlin. [11] [28]

## 2.2.3 Aplicación híbrida

Las aplicaciones híbridas son aplicaciones móviles desarrolladas con tecnologías web y compiladas para cada sistema operativo del dispositivo inteligente. No se deben confundir con las aplicaciones PWA ya que, las aplicaciones híbridas deben ser compiladas y descargadas mediante las plataformas de distribución digital de los dispositivos inteligentes, y las aplicaciones PWA se pueden ejecutar sin necesidad de ser instaladas en el dispositivo, ya que se crean accesos directos a los navegadores web (sin mostrar la apariencia de un navegador). [28]



## 2.3 Framework de frontend

En la actualidad existen multitud de *frameworks* para implementar el cliente de una aplicación basada en HTML, CSS y Javascript. A continuación, se realizará una descripción de los más importantes.

### 2.3.1 Angular

Angular es un *framework* para el desarrollo de aplicaciones basado en Typescript y mantenido por Google que se lanzó en el año 2016. Se basa en el modelo Modelo-Vista-Controlador MVC mediante *templates* que contienen el HTML, con etiquetas especiales de Angular, para formar la vista, componentes que gestionan estas *templates* para formar el controlador, y servicios que encapsulan la lógica de la aplicación para generar el modelo. Estos componentes y servicios se organizan dentro de módulos. [12]

Por otro lado, Angular utiliza promesas para la comunicación con el *backend* y obtención de sus datos desde el modelo de la aplicación; además se suele utilizar junto con Bootstrap, una librería para la creación de interfaces gráficas mediante material design. Para finalizar, hay que destacar que Angular suele utilizarse en aplicaciones web cliente-servidor, donde el servidor es implementado con Spring.[13]

### 2.3.2 React.js

Este *framework* basado en Javascript se lanzó en 2013 con el respaldo de Facebook y su principal virtud es la facilidad de creación de aplicaciones SPA. React.js se basa en un modelo Modelo-Vista-Vista-Modelo MVVM, aunque también puede tener como modelo MVC, mediante componentes interactivos y reutilizables para la creación de la interfaz del usuario. [14]

La principal característica de React.js es que está construido en torno a la creación de funciones que actualizan la interfaz de usuario cuando existe una actualización del estado de la aplicación. Esto se realiza mediante un algoritmo que identifica las diferencias entre el estado actual de la aplicación y los cambios que se aplican y actualiza tanto la interfaz gráfica como el estado. Actualmente, Facebook e Instagram están desarrolladas con este *framework* en su parte *frontend*.

### 2.3.3 Vue.js + Nuxt.js

Vue.js es un *framework* basado en Javascript y en un modelo MVVM para construir interfaces y aplicaciones SPA. Su fecha de lanzamiento fue en 2014 y se asemeja a React.js en que usa componentes interactivos y reutilizables, así como el uso de estados reactivos, aunque aborda esta problemática de forma distinta. La principal diferencia de Vue.js con React.js es que el primero usa plantillas escritas en HTML y el segundo usa plantillas derivadas de Javascript: JSX, aunque también es posible realizarlo con HTML, pero pierde eficiencia en su algoritmo. Esto produce que la curva de aprendizaje sea mucho más sencilla en Vue.js que con React.js. [15] [16]

Por su parte, Nuxt.js es un *framework* derivado de Vue.js utilizando las librerías oficiales de este, como son Vuex.js, vue-route.js o Vuetify.js y otras herramientas de desarrollo como webpack o BABEL haciendo que el desarrollo de las aplicaciones sea más potente y eficaz que Vue.js. Nuxt.js aplica una arquitectura de proyecto más sólida que un proyecto Vue.js, haciendo que se considere como la columna vertebral de un proyecto Vue.js. [17]

La unidad mínima de Vue.js es el *component* que está compuesto de tres fragmentos de código: Uno de HTML que corresponde a las plantillas y la interfaz gráfica del usuario, otro de Javascript para aplicar lógica a esta plantilla y, por último, un fragmento de CSS para aplicar estilo a la plantilla. Este componente tiene un ciclo de vida que abarca desde su creación hasta su nula utilización. [18]

Vue.js posee una serie de librerías oficiales para realizar una aplicación mucho más robusta. En primer lugar, hay que destacar Vuex.js, que es el patrón de gestión de estados, similar a Redux.js de React.js o Angular, que permite centralizar el estado de la aplicación y aplicar reglas (funciones) para asegurar que la mutación de este se hace de forma predecible. Otra librería que destacar es Vuetify.js, que se asemeja a Bootstrap para Angular, gracias a estas librerías se pueden crear las plantillas con estilo de material design de forma *responsive*. Para finalizar, otra librería que se utiliza junto con Vue.js es vue-route, que permite realizar el enrutamiento de las páginas de la aplicación de forma automática en función del lugar donde se encuentran estas dentro del proyecto. [19] [20]

## 2.4 Arquitectura backend

En este punto nos vamos a centrar en las distintas alternativas existentes para la persistencia de los datos y el *host* del servidor.

### 2.4.1 Persistencia

Actualmente existen dos modelos de gestión de bases de datos: modelo relacional y modelo no relacional. El primer modelo, también se conoce como bases de datos SQL, ya que es el lenguaje utilizado para implementar dicho modelo, se basa en la creación de tablas (relaciones) compuestas por filas, que se consideran registros, y columnas, que se corresponden con los datos. Cada dato tiene una clave primaria que le identifica dentro de la tabla y gracias a esta, se pueden realizar todas las relaciones de datos entre distintas tablas.

Actualmente, las bases de datos SQL son el modelo más utilizado en la industria informática debido a su rapidez de consulta, ya que se basa en el álgebra y cálculo relacional con el fin de realizar estas consultas de forma sencilla. También ayuda a la comprensión de las bases de datos para los programadores. Por otro lado, la principal desventaja que tienen estas de bases de datos es la escalabilidad horizontal y la poca flexibilidad debido a los requisitos estrictos de la estructura que presenta. Oracle, MySQL, PostgreSQL o MariaSQL son algunos de los ejemplos de las bases de datos relacionales que actualmente existen en el mercado. [21] [22]

Por otro lado, las bases de datos no relacionales, también conocidas como bases de datos NoSQL, se caracterizan por no guardar datos en tablas, si no, por ejemplo, en colecciones de documentos, donde estos pueden ser de cualquier tipo, sin seguir ninguna regla marcada. Por lo general, estos documentos son archivos JSON que suelen contener un identificador que le identifica dentro de la colección. [23]

La principal ventaja de las bases de datos NoSQL, es que tiene una estructura de sistema distribuido, lo que provoca mayor escalabilidad horizontal y una mayor velocidad de escritura. Pero en contra, no hay establecido un soporte común para este tipo de bases de datos, lo que provoca menor fiabilidad y coherencia en los datos. [24]

Por lo general, este tipo de bases de datos se suele utilizar cuando existen una gran cantidad de datos y las consultas son sencillas. Actualmente, existen bases de datos NoSQL que buscan un equilibrio entre la lectura y escritura de los datos, pero se pierde la idea de almacenamiento rápido de los datos. Algunos de los ejemplos de este tipo de bases de datos son MongoDB o Redis.

### 2.4.2 Hosting

Actualmente, los servidores se pueden albergar en máquinas propias o en la nube. El *host* en la nube consiste en máquinas que una empresa tecnológica pone a disposición de la gente y estos despliegan sus desarrollos, pero no tienen control de las máquinas.

Normalmente, existe un coste si un usuario tiene su propio servidor, ya que necesita comprar el material necesario para crear el servidor, realizar un mantenimiento, así como los costes de luz y e internet. A su vez, los servidores en la nube también tienen un coste asociado al número de peticiones procesadas o los datos guardados, por lo que el coste suele ser bastante similar entre ambas opciones.

Actualmente, existen varias compañías que permiten la creación de servidores en la nube y que, normalmente, si el tráfico no supera unos límites estipulados son gratuitas. Algunos de estos productos son:

- **Amazon Web Services.** AWS es la plataforma que Amazon para que los usuarios puedan desplegar sus servidores en la nube. Normalmente, obligan a la creación de una API REST con una serie de *endpoints* para la comunicación cliente-servidor. En cuanto a la persistencia, AWS nos permite elegir entre bases de datos SQL o NoSQL. [25]
- **Firebase.** Firebase es una plataforma para el desarrollo de aplicaciones en la nube gestionada por Google y consta de un conjunto de servicios y *hosts* por todo el mundo para poder albergar la aplicación. Los principales servicios que proporciona son *Firebase Authentication*, *Cloud Firestore*, *Firebase Storage* y *Firebase Functions*. La base de datos de esta plataforma es una base de datos NoSQL basada en colecciones y documentos de clave-valor. Todos estos servicios son llamadas desde el cliente sin necesidad de pasar por un *backend* como tal. [26]

Gracias a esta computación en la nube se permite crear una arquitectura *backend* denominada *backend as a services*, que consiste en vincular las aplicaciones con el almacenamiento en la nube, servicios analíticos, gestión de usuarios o la integración con redes sociales a través de las distintas API que proporcionan las distintas plataformas. [27]

### 2.4.3 Backendless

El *backendless* es la evolución natural del *backend as a services* y consiste en reducir al mínimo la computación en el backend y que el cliente se encargue de llamar a los distintos servicios que las plataformas nos ofrecen para crear el *backend* de la aplicación sin necesidad de desarrollar peticiones REST a *endpoint* del servidor, excepto alguna funcionalidad concreta que sea necesario la ejecución en el lado del servidor.

Con esta arquitectura no se busca la eliminación del *backend*, ya que este sigue existiendo mediante los distintos servicios dentro de la plataforma, lo que se busca es el no desarrollo de este y aprovechar al máximo estos servicios y así realizar aplicaciones mucho más ligeras en cuanto a computación, ayudando a la mantenibilidad.



## 3 Análisis

---

Este capítulo está centrado en la fase de Análisis del proyecto. Se observarán las distintas reuniones realizadas con el cliente, para continuar con los requisitos obtenidos de estas y finalizar con los casos de uso y una descripción formal del sistema.

### 3.1 Reunión con el cliente

Dado que este TFG ha sido enfocado como un proyecto real, en el que un cliente se cita con un consultor para exponer el problema y el consultor encuentre la solución ideal que lo resuelva, en este apartado se va a exponer las primeras reuniones que el autor del TFG ha mantenido con un grupo de pequeñas empresas y trabajadores autónomos para recoger las distintas problemáticas y, así poder realizar una lista de requisitos y una definición del sistema que se va a desarrollar. Dado que es un grupo, acordaron tener un representante para tratar todos los temas y asistir a todas las reuniones con el consultor.

En estas primeras reuniones, el representante explicó el problema que existía y las necesidades de tener una plataforma donde poder publicitarse y vender los productos de cada comercio con un bajo coste, de fácil mantenimiento y usabilidad. En este punto, el consultor propuso una serie de sugerencias a cerca del tipo de aplicación que se desea implementar y distintas funcionalidades extra que podían ser interesantes para este proyecto.

Con la idea clara, el autor de este TFG contacta con el tutor de este con el fin de exponer la idea de realizar este proyecto como TFG. El tutor indica que es un proyecto con un alto grado de trabajo, pero que se puede realizar un MVP de esta plataforma como TFG y que, en trabajos futuros, se puede realizar la finalización. Tras llegar a un acuerdo, el tutor acuerda la asignación de este TFG.

De estas reuniones, el autor del TFG se compromete a realizar un análisis detallado de toda la información recogida y un diseño de la aplicación completa.

### 3.2 Descripción de la aplicación

En este apartado se realiza una descripción de la funcionalidad y características que debe tener este proyecto una vez finalizadas todas las iteraciones. Se trata de una aplicación web multiplataforma en la que los comercios y trabajadores autónomos tendrán un espacio individual para darse a conocer dentro de esta plataforma. Además, se incluirá una plataforma de comercio online donde estos comercios y autónomos puedan incluir su propio inventario pudiendo vender lo que se desee dentro de una plataforma estable y con distintos comercios fomentando las ventas.

Una de las características más relevantes es la posibilidad de que en el espacio individual de cada comercio o autónomo sólo se muestre el inventario de este, simulando así que se tiene una web propia, aunque sea dentro de esta plataforma, pero siendo accesible desde cualquier sitio sin necesidad de tener que ser desde el principio de la aplicación. Los espacios personales no serán personalizables, aunque sí se podrán incluir imágenes e información propia de cada comercio para tener todo lo necesario para fomentar sus ventas y darse a conocer en internet.

Esta aplicación constará con distintos tipos de usuarios que podrán realizar unas acciones u otras. En primer lugar, existirán los *empleados* y *administradores*, encargados de la gestión de los comercios registrados, destacar que el empleado tendrá menos permisos que el administrador. Por otro lado, existirá el *usuario*, que tendrá un perfil dentro de la aplicación y podrá realizar pedidos a los comercios registrados. Para finalizar, existirá un perfil de *superusuario* o *superadministrador* encargado de gestionar toda la aplicación.

### **3.3 Análisis de requisitos**

En este punto se van a exponer una lista de los requisitos funcionales y no funcionales obtenidos de las reuniones con la representante de los clientes y de la descripción de la aplicación.

#### **3.3.1 Requisitos funcionales**

##### **3.3.1.1 Autenticación**

**RF1.** Cualquier miembro registrado en la aplicación podrá iniciar sesión en el sistema.

- Este acceso se realizará mediante un correo electrónico y una contraseña elegida por el usuario, o mediante una cuenta de Google.
- En el caso de ser un empleado, el primer acceso se realizará con una contraseña generada por el sistema a la hora del registro y posteriormente el usuario la podrá modificar en un futuro.
- En función del rol asociado a la aplicación (Superadministrador, administrador, empleado y usuario) se mostrará una vista distinta de la aplicación.

**RF2.** Cualquier usuario registrado podrá cerrar sesión del sistema en cualquier momento.

- Esta acción se realizará mediante un botón.
- Todos los cierres de sesión realizados, independientemente del perfil de usuario, llevarán a la ventana principal de la aplicación.

##### **3.3.1.2 Gestión del comercio**

**RF3.** Cualquier persona con un comercio puede registrar este en la aplicación.

- Esta persona debe estar ya registrada en la aplicación.
- El registro se realizará en el botón de la pantalla principal y seleccionando la opción de registrar comercio.
- Será necesario incluir datos del comercio como Nombre, Logotipo, Información General, lugar de tienda física (si existiese), fotos varias y datos de contacto.
- El usuario que registre el comercio será considerado administrador de este.
- El rol superusuario puede registrar también comercios si dispone de los datos necesarios para realizar esta acción.

**RF4.** El administrador de un comercio puede dar de baja un comercio.

- Esta acción se realizará en el perfil del comercio.
- Al dar de baja el comercio, desaparecerán todos los productos de esta en la aplicación.
- Serán eliminados todos los empleados asociados a este comercio.
- El rol superusuario también puede dar de baja un comercio.

**RF5.** El superusuario puede bloquear a un comercio si este no cumple con las políticas de la aplicación.

- En este caso, el comercio pasa a un estado de suspensión.
- No se mostrarán los artículos a la venta de este comercio.
- No se eliminará ningún dato.
- Se comunicará mediante email el motivo del bloqueo.
- Se bloqueará el acceso a todos los miembros relacionados con ese comercio.

**RF6.** El administrador del comercio puede modificar los datos de este.

- Esta modificación se realizará en el perfil del comercio.
- Los posibles datos que modificar son el Logotipo, información sobre el comercio, localización de la tienda física.

**RF7.** Cualquier usuario puede visualizar el espacio individual de un comercio.

- En esta página, se visualizará la información del comercio como descripción, imágenes, localización o datos de contacto.
- Se mostrarán todos los artículos de este comercio.

**RF8.** Cualquier usuario registrado puede dar una valoración del comercio.

- Esta valoración se realizará en el espacio individual del comercio.
- La valoración irá con un rango de 0 a 5.

**RF9.** Cualquier usuario relacionado con el comercio puede visualizar el perfil de este.

- Este perfil es una vista individual y no es la que el resto de los usuarios visualizan.
- Desde esta vista se realizarán distintas acciones como eliminar o modificar.
- El superusuario también tiene acceso a este perfil.

### **3.3.1.3 Gestión de empleados**

**RF10.** Cualquier administrador de un comercio registrado puede registrar a usuarios con el rol de empleado.

- Esta acción se realizará en la lista de empleados visible solo por el administrador.
- Será necesario incluir un correo del empleado, así como diversos datos personales de este.
- Se generará una contraseña automática para el primer acceso.
- El empleado registrado quedará relacionado con el comercio.

**RF11.** El administrador de un comercio puede dar de baja a un empleado incluido en su comercio.

- Esta acción se realizará en la lista de empleados visible solo por el administrador.
- Una vez dado de baja un empleado, este no tiene acceso a la aplicación.

**RF12.** El administrador de un comercio puede bloquear a un empleado incluido en su comercio.

- Esta acción se realizará en la lista de empleados visible solo por el administrador.
- Una vez bloqueado un empleado, este no tiene acceso a la aplicación.
- El empleado pasa a un estado de suspensión.
- No se eliminará ningún dato.
- Se comunicará mediante email el motivo del bloqueo.

**RF13.** Cualquier empleado registrado puede modificar su contraseña.

- Esta acción se realizará mediante un formulario accesible por el correo que le llegue a este cuando se realiza el registro.
- También se podrá modificar la contraseña en el perfil personal del empleado.

**RF14.** Cualquier empleado registrado puede visualizar su información personal.

- Es una vista personal donde visualizar los datos con los que ha sido registrado.
- El superusuario de la aplicación tiene acceso a esta vista.

### **3.3.1.4 Gestión de usuarios**

**RF15.** Cualquier persona puede registrarse en la aplicación para realizar compras.

- El registro se realizará desde la pantalla principal.
- Será necesario incluir un correo electrónico y una contraseña, así como diversos datos personales como nombre completo, teléfono de contacto o dirección.

**RF16.** Cualquier usuario registrado puede darse de baja en la aplicación.

**RF17.** El superusuario de la aplicación puede bloquear a un usuario.

- El usuario pasa a un estado de suspensión.
- No se eliminará ningún dato.
- Se comunicará mediante email el motivo del bloqueo.
- Si el usuario tiene un rol administrador, el comercio al que pertenece quedarán también bloqueados y, a su vez, todos sus empleados.

**RF18.** Cualquier usuario registrado puede visualizar sus datos personales.

- Es una vista personal donde visualizar los datos con los que ha sido registrado.
- El superusuario de la aplicación tiene acceso al perfil.

**RF19.** Cualquier usuario registrado puede modificar su contraseña.

- Se realizará vía formulario accesible desde la vista del perfil del usuario.
- El superusuario podrá enviar un mail con el formulario si el usuario en cuestión contacta con él.

**RF20.** Cualquier usuario registrado puede modificar sus datos personales.

- Se realizará vía formulario accesible desde la vista del perfil del usuario.
- El superusuario de la aplicación también podrá realizar modificaciones en el perfil con el consentimiento del usuario.

### **3.3.1.5 Artículos**

**RF21.** Los usuarios con rol administrador o empleado relacionados con un comercio pueden dar de alta artículos dentro de este.

- Esta acción se realizará mediante un formulario donde habrá que indicar: nombre, descripción, código de barras, precio y cantidad.
- Para llegar al formulario, se realizará sobre la vista del inventario que tendrán los administradores y empleados.
- Los artículos tendrán una relación con el comercio desde donde han sido dados de alta.

**RF22.** Los usuarios con rol administrador o empleado relacionados con un comercio pueden aplicar descuentos a los artículos.

- Se realizará mediante la vista del inventario.
- Se incluirá un porcentaje de descuento que se guardará junto con los datos del artículo.

**RF23.** Los usuarios con rol administrador o empleado relacionados con un comercio pueden dar de baja un artículo.

- El artículo desaparecerá del inventario del comercio.
- El artículo desaparecerá del inventario completo de la aplicación.

**RF24.** Cualquier persona puede visualizar los artículos existentes en la aplicación.

- Los artículos son de todos los comercios existentes en la aplicación.
- En una primera búsqueda se mostrará el título, el precio y una imagen del artículo.
- También existirá una ventana propia para cada artículo.

**RF25.** Cualquier persona puede realizar una búsqueda de los artículos.

- Esta acción se realizará desde la página principal de la aplicación o desde el espacio individual de cada comercio.
- La búsqueda se realizará por palabras clave en el título del artículo.
- El resultado de esta búsqueda serán los artículos que coincidan con lo escrito en el buscador.

**RF26.** Cualquier persona puede realizar un filtrado de todos los artículos.

- Esta acción se realizará desde la página principal de la aplicación o desde el espacio individual de cada comercio.
- El filtrado de los artículos puede ser por rango de precios o categorías.
- El resultado es un conjunto de productos a los que se les aplique el filtro.

**RF27.** Cualquier persona puede organizar los artículos.

- Esta acción se realizará desde la página principal de la aplicación o desde el espacio individual de cada comercio.
- Se pueden ordenar en orden alfabético y por precio.

**RF28.** Cualquier usuario autenticado puede incluir un artículo al carrito de la compra.

- Esta acción se realizará o desde la vista de todos los artículos o desde la vista individual del artículo.
- Se podrá decidir la cantidad de artículos que se desean incluir.

**RF29.** Cualquier usuario autenticado puede incluir comentarios en los artículos.

- Esta acción se realizará desde la vista individual del artículo.
- El comentario constará con un título y un cuerpo de comentario.

**RF30.** Cualquier usuario autenticado puede dar una valoración del artículo.

- Esta acción se realizará desde la vista individual del artículo.
- La valoración irá de 0 a 5.



### **3.3.1.6 Pedidos y envíos**

**RF31.** Cualquier administrador del comercio puede añadir un nuevo tipo de envío al comercio.

- Se realizará mediante la vista de métodos de envío solo accesible para el administrador.
- Se realizará vía formulario donde se incluirá nombre de la compañía y precio del transporte.

**RF32.** Cualquier administrador del comercio puede eliminar un tipo de envío del comercio.

- Se realizará mediante la vista de métodos de envío solo accesible para el administrador.

**RF33.** Cualquier usuario autenticado puede realizar un pedido.

- Esta acción se realizará pagando y finalizando el carrito de la compra.
- Todos los productos existentes en el carrito pasarán a formar parte del pedido.
- Como pueden existir artículos de distintos comercios, se realizará un pedido por comercio existente.
- Cuando se finalice el pedido, la cantidad del inventario se actualizará.
- Se enviará un mail al usuario explicando el pedido realizado.

**RF34.** Cualquier usuario autenticado puede seleccionar el tipo de envío que desea para el pedido.

- Esta acción se realizará cuando el usuario realice un pedido.
- Se mostrarán las opciones de los distintos métodos de envío relacionados con el comercio.
- El precio del envío conjunto se sumarán al precio final.

**RF35.** La gestión de los envíos está externalizada.

- Una vez se seleccione el método de envío y se finalice el pedido, la aplicación se conectará con el proveedor de envío para generar la orden.

**RF36.** Cualquier usuario registrado puede visualizar el estado de los pedidos (envíos) que tiene realizados en la aplicación.

- Esta acción se realizará en la vista de pedidos del usuario.

**RF37.** Los usuarios con rol administrador o empleado relacionados con un comercio puede modificar el estado del pedido.

- Esta acción se realizará en la vista de pedidos del comercio.

**RF38.** Cualquier usuario registrado puede cancelar el pedido en un periodo de 24h desde que se realizó el pedido.

- Esta acción se realizará en la vista de pedidos del usuario.
- Esto no supondrá ningún coste para el cliente.
- Al realizar esta acción, se devolverá el dinero al usuario.
- Al realizar esta acción, se actualizará el inventario.

**RF39.** Cualquier usuario autenticado podrá pagar un pedido.

- Los métodos de pago son vía tarjeta de crédito o PayPal.
- Se realizará mediante una pasarela externa a la aplicación.
- Esta acción será el último paso para finalizar un pedido.

**RF40.** Cualquier usuario registrado puede visualizar el carrito de la compra actual.

- Se visualiza una lista de los artículos que contiene, cantidad y precio.
- Se visualiza el precio total del carrito.
- Se da la opción de realizar el pedido.

**RF41.** Cualquier usuario registrado puede eliminar artículos del carrito de la compra.

- Esta acción se realizará mediante la vista del carrito.

**RF42.** Cualquier usuario registrado puede modificar la cantidad los artículos que dispone en el carrito de la compra.

- Esta acción se realizará mediante la vista del carrito.
- La cantidad nunca puede ser inferior a 1.
- La cantidad no tiene límite superior, pero si se supera a las existencias en el inventario, saltará un error al finalizar el pedido.

### 3.3.2 Requisitos no funcionales

**RNF1.** Aplicación web multiplataforma.

**RNF2.** Posibilidad de ser utilizada como aplicación nativa.

**RNF3.** Mantener la privacidad y seguridad de los datos de los usuarios.

**RNF4.** El borrado o bloqueo de algún documento (usuario, comercio, artículo) se realizará de forma lógica, dejando persistente los datos, pero no se mostrarán a los usuarios finales.

**RNF5.** Guardar los datos de cada usuario registrado en base de datos noSQL en la nube.

**RNF6.** Se debe usar la pasarela de pago externa para todo tipo de pagos.

**RNF7.** El sistema debe proporcionar mensajes de error informativos y orientados al usuario final.

**RNF8.** Tiempo de respuesta a cada acción en el sistema menor a nueve segundos.

**RNF9.** Contenido estructurado, claro, conciso y con un diseño simple.

**RNF10.** El sistema soportará el uso de centenares de usuarios simultáneamente.

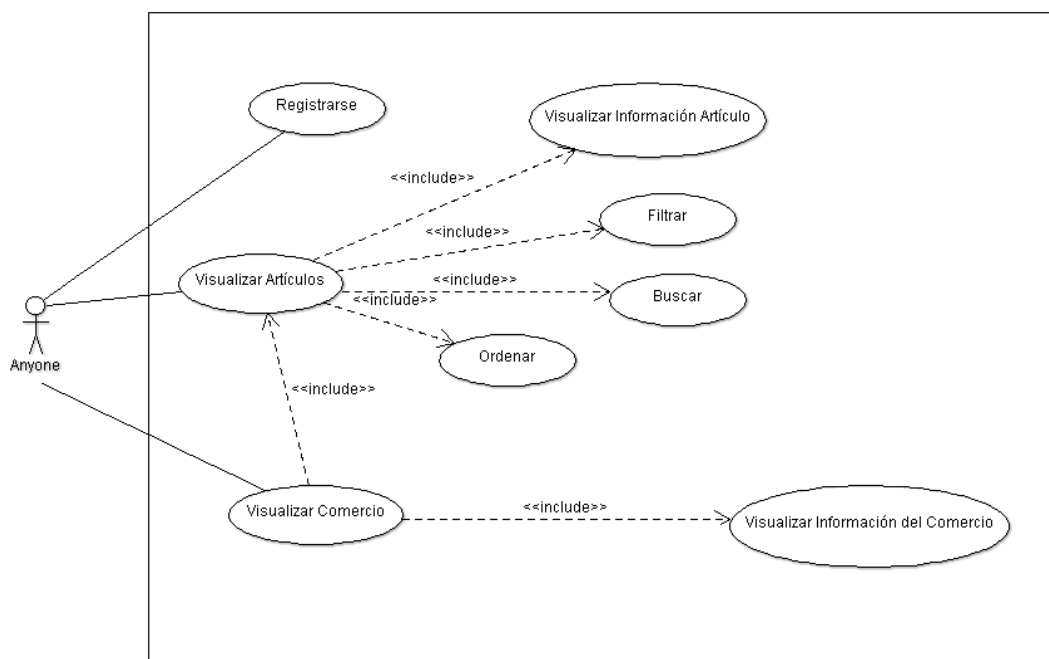
**RNF11.** La aplicación debe visualizarse tanto en móviles como en navegadores de forma clara.

**RNF12.** La aplicación deberá ser visible en los navegadores con 5% superior en cuota de mercado (Chrome, Firefox, Safari)

### 3.4 Casos de usos

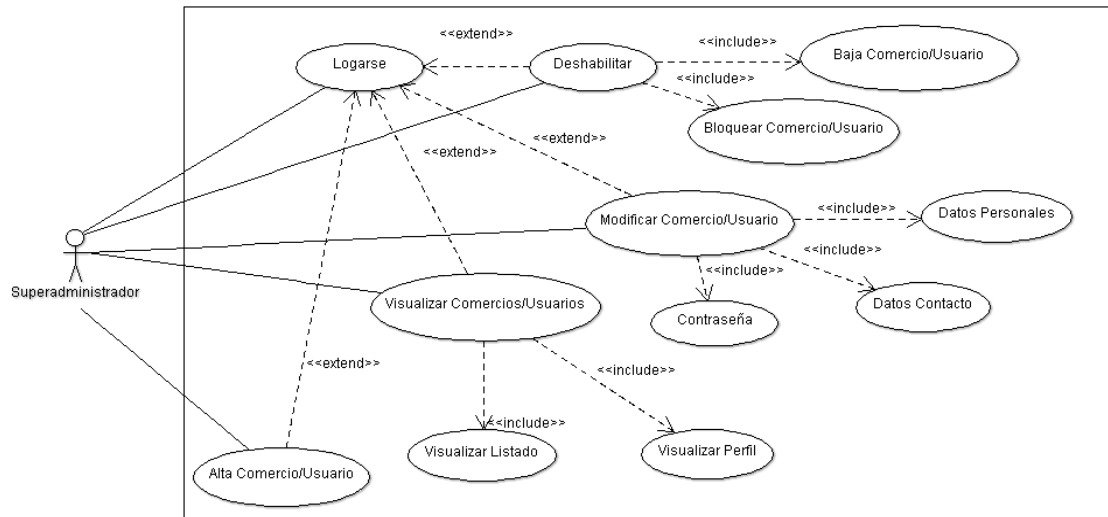
En este apartado se van a mostrar los diagramas de casos de uso divididos en función de los roles que interactúan en la aplicación. Se ha realizado de esta forma con el fin de obtener diagramas más pequeños y fáciles de leer.

En la figura 1 se muestra el diagrama de casos de uso de cualquier usuario sin rol que accede a la aplicación. Como se observa, la funcionalidad está limitada a visualizar los artículos y comercios y el posible registro si lo desea.



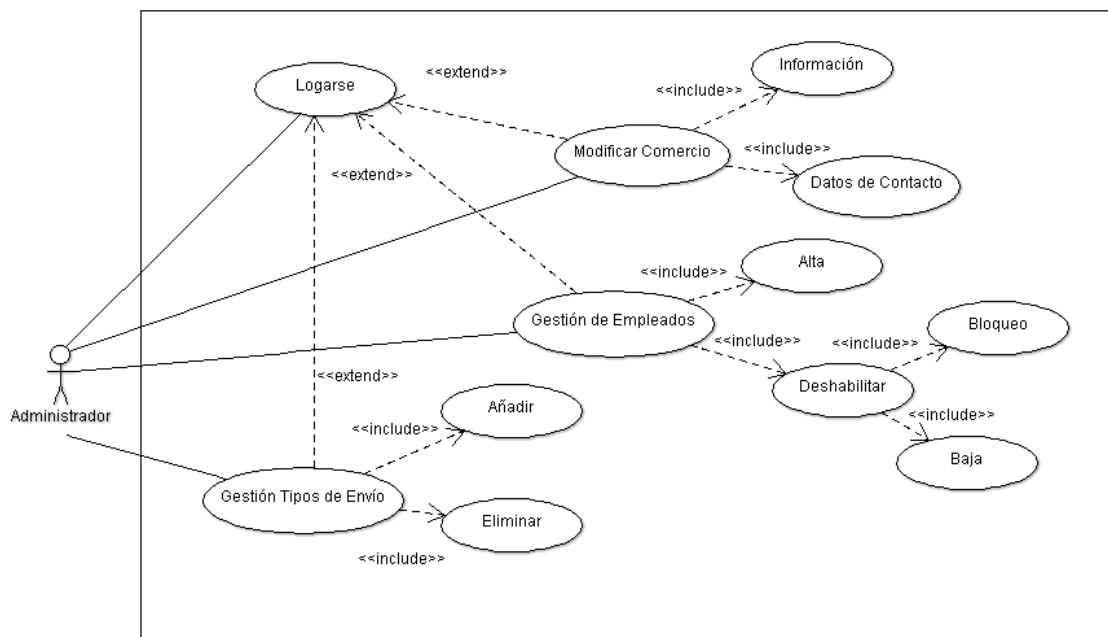
**Figura 1. Diagrama de Casos de Uso Usuario sin Rol**

En la siguiente figura, se muestra un usuario con el rol de *Superusuario*, encargado de gestionar la aplicación a partir de una serie de vistas donde podrá deshabilitar comercios, usuarios, así como visualizar a todos ellos.

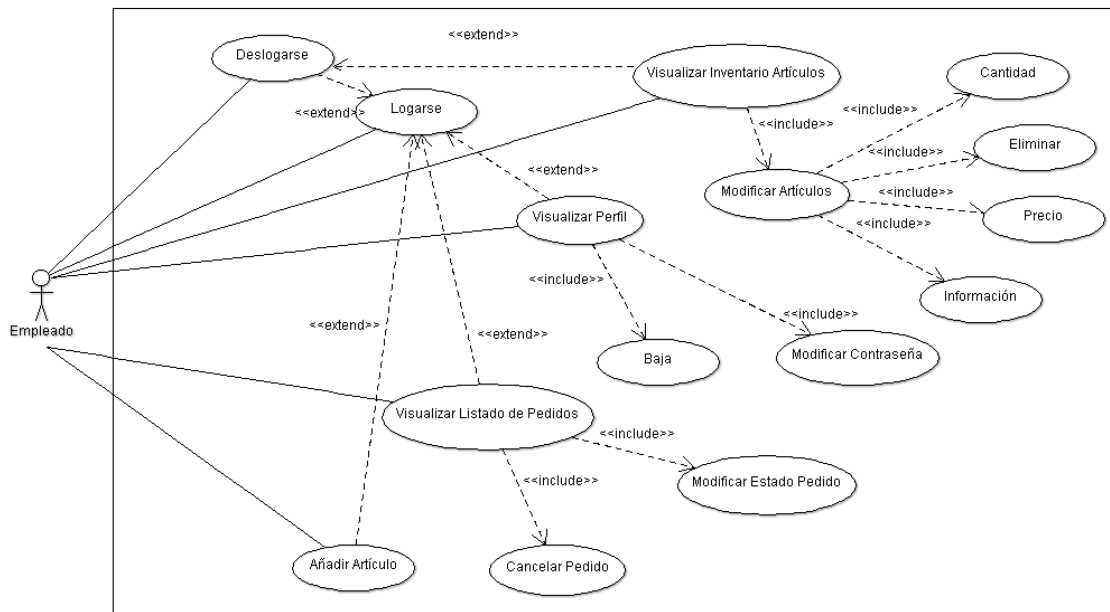


**Figura 2. Diagrama de Casos de Uso Usuario con Rol Superusuario.**

En la figura 3, se observa el caso de uso de un usuario *administrador*, que es el administrador/gerente de los comercios existentes en la aplicación. Cabe destacar, que a parte de la funcionalidad indicada y, debido a la legibilidad de esta figura, también incluye todos los casos de uso de la figura 4, que se corresponde a un rol *empleado*.

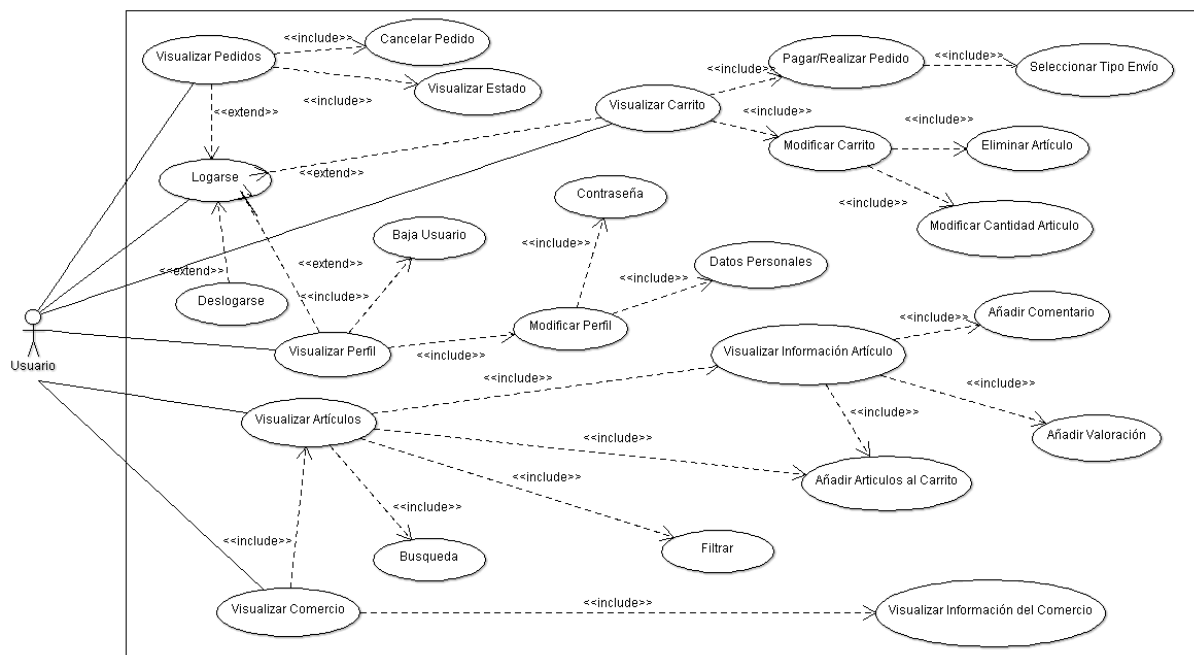


**Figura 3. Diagrama de Casos de Uso Usuario con Rol Administrador.**



**Figura 4. Diagrama de Casos de Uso Usuario con Rol Empleado.**

La figura 5 corresponde a los casos de uso correspondientes a un *usuario* registrado en el sistema. En ella se puede observar que se incluyen todos los casos de uso correspondientes a un usuario sin registrar y, además, todo lo relacionado con su perfil y la funcionalidad de realizar pedidos.



**Figura 5. Diagrama de Casos de Uso Usuario con Rol Usuario.**

Para finalizar, hay que indicar que en el Anexo A se encuentran definidos una serie de casos de uso de la aplicación realizados en conjunto con estos diagramas vistos anteriormente.

## 4 Diseño

En este capítulo se va a exponer el diseño del MVP a desarrollar durante el transcurso del proyecto. Se observarán las tecnologías usadas y su motivo de su uso, así como la arquitectura de la aplicación y las maquetas realizadas con el fin de observar un primer diseño de la aplicación.

### 4.1 Tecnologías y lenguajes empleados

Tras observar las distintas alternativas en el capítulo 2 de este documento, se ha decidido implementar una aplicación web PWA, ya que nos permite crear una aplicación que puede simular una aplicación nativa sin necesidad de tener que desplegar la aplicación en las plataformas de distribución de aplicaciones teniendo en cuenta los distintos sistemas operativos de cada dispositivo inteligente.

En cuanto al *framework* a utilizar, se ha decidido realizar el proyecto con Nuxt.js, ya que es una de las tecnologías más punteras en el desarrollo web en el momento de la realización de dicho proyecto, así como permitir que la carga de procesamiento de datos esté en el cliente pudiendo implementar arquitectura *backendless* y permitir crear aplicaciones PWA de una forma sencilla. Pero la principal decisión del uso de Nuxt.js frente a React.js es la curva de aprendizaje; frente a Angular.js es que con este *framework* se puede utilizar un patrón MVVM. Además, se van a utilizar una serie de librerías oficiales de Vue.js ya incluidas en un proyecto Nuxt.js como son Vuetify.js, vue-route.js o Vuex.js.

Para finalizar, el *backend*, como se ha comentado anteriormente se va a seguir una estructura *backendless* apoyándonos en Firebase de Google que nos permite utilizar servicios como *Firebase Authentication*, *Cloud Firestore* y *Firebase Functions*. La principal elección de Firebase sobre AWS es que la primera nos permite crear un *backendless* puro de procesamiento en la nube sin necesidad de crear *endpoints* si no son necesarios, y AWS nos obliga a crear una primera capa de acceso al API, aunque todo su procesamiento esté en la nube.

### 4.2 Maquetas

Las maquetas realizadas sobre la interfaz gráfica de la aplicación web se encuentran en el Anexo B de este documento. Estas maquetas son una primera vista de lo que se presenta al cliente y este acepta para su posterior implementación.

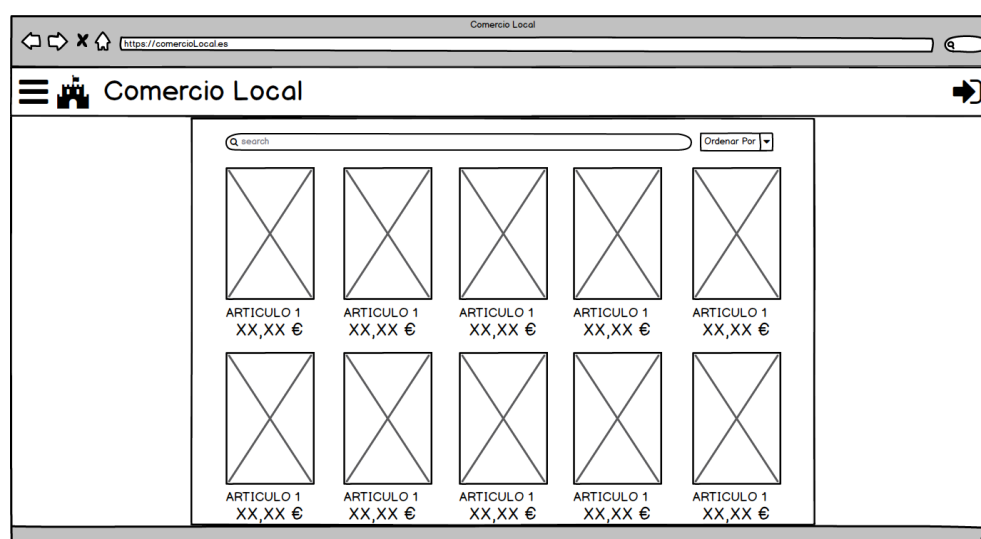


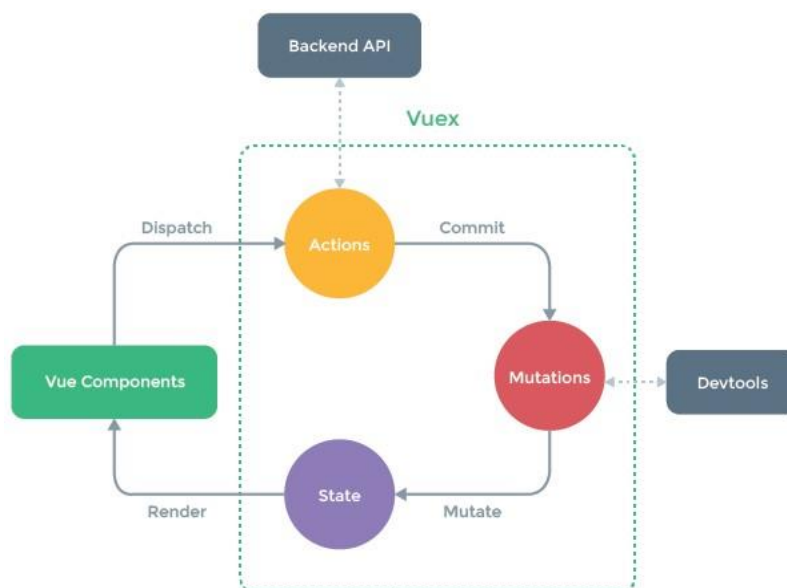
Figura 6. Maqueta de la vista inicial de la aplicación.

## 4.3 Arquitectura de la aplicación

En este apartado se va a hacer un diseño más exhaustivo de la arquitectura de la aplicación una vez definidas las tecnologías que se van a utilizar.

### 4.3.1 Frontend

Como se ha comentado en el apartado 4.1, el *framework* utilizado para el desarrollo del cliente de la aplicación será Nuxt.js, que sigue un modelo MVVM por lo que, en este punto, se va a explicar dicho modelo. Para realizar este modelo, Nuxt.js utiliza la librería Vuex.js, basado en el patrón de estados Flux.js, que nos permite tener aplicaciones reactivas por parte de los componentes en función del estado de la aplicación.



**Figura 7. Integración de MVVM con Flux.js.**

Comenzando con la vista, esta se va a realizar a partir de componentes, que formarán distintas páginas en función de la sesión existente en la aplicación en ese momento. Estos componentes serán creados con la ayuda de la librería Vuetify.js, que nos ayudará a crear una interfaz de usuario agradable para este siguiendo las maquetas realizadas en el apartado 4.2. Las páginas y el enrutamiento de estas se realizarán con la ayuda de la librería vue-route.js. Hay que destacar que las páginas tienen una relación muy cercana a los documentos guardados en bases de datos y explicados en el punto 4.3.3 de este documento con el fin de optimizar la carga de datos en las páginas.

Centrándonos ya más en el propio estado, Vuex.js nos da un *store* que contendrá el estado de la aplicación y una serie de funciones para modificar dicho estado. En primer lugar, un componente modificará el estado de la aplicación a partir *actions* (acciones), que se encargarán de modificar la base de datos y, por otro lado, mediante una mutación, modificar el propio estado. Cuando el componente Vue.js observe que hay cambios en el estado, actualizará sus datos y mostrará la nueva información.

También es posible realizar modificaciones del estado de la aplicación a partir de un componente mediante *mutations* (mutaciones) pero, por lo general y en particular de esta aplicación, todas las modificaciones de estado se realizarán mediante acciones, ya que se modificará en base de datos primero y posteriormente se actualizará el estado. Por otro lado, para recuperar los datos y modificaciones del estado por parte de los componentes, se realizará a través de *getters*, ya que se aplicarán modificaciones de estos antes de poder renderizar la vista.

### 4.3.2 Backend

Como se ha comentado en el apartado 4.1, el servidor va a seguir una arquitectura *backendless* basado en Firebase, que nos proporciona una serie de servicios y un *host* en la nube para alojar la aplicación.

Cuando se crea un proyecto en Firebase, este nos indica en qué servidor deseamos alojar este entre Europa, Estados Unidos o Asia y, desde ese momento, ya tenemos un espacio reservado para la aplicación. En este espacio, podemos utilizar los servicios Firebase que nos proporciona para la gestión del proyecto. En este caso, los servicios serán:

- **Firebase Authentication.** Este servicio nos proporcionará la gestión de los usuarios de la sesión como es el registro, el inicio y cierre de sesión a partir de un correo electrónico y una contraseña.
- **Cloud Firestore.** Este servicio proporciona una base de datos NoSQL para la persistencia de la aplicación. En el punto 4.3.3 se entrará más en detalle sobre el modelo de datos.
- **Firebase Functions.** Con este servicio conseguimos tener *endpoints* propios y funcionalidad corriendo en el servidor. Estas funciones se ejecutan en el espacio reservado en el servidor de la nube que hayamos indicado y son útiles cuando hay que realizar alguna funcionalidad extra que no nos proporciona el resto de servicios Firebase. Para este proyecto, será necesario crear algún *endpoint* para la conexión con terceros como puede ser la pasarela de pago externa.

La comunicación cliente-servidor se realizará mediante las acciones del *store* de Vuex.js, por lo que es necesario la creación de reglas de seguridad con el fin de mantener la robustez de la aplicación. Firebase nos permite crear reglas de lectura y escritura que, en este proyecto, serán generadas a partir los datos de la sesión.

### 4.3.3 Persistencia

Como se ha comentado en el punto anterior, se va a utilizar la herramienta *Cloud Firestore* para la persistencia de los datos de la aplicación. Este API se basa en un modelo de base de datos NoSQL cuyos componentes son *colecciones* y *documentos*.

Una colección es un conjunto clave valor de documentos, donde la clave es el id generado por Firebase o indicado por el desarrollador y el valor es un JSON con la información deseada. La principal característica de las colecciones es que pueden contener todo tipos de documentos, no es necesario que todos los documentos contengan la misma información. Por otro lado, hay que destacar que un documento, a su vez, puede contener otras colecciones, pero no otros documentos, aunque es una práctica no aconsejada debido a que aplica una gran complejidad en el desarrollo y la mantenibilidad.

En cuanto el diseño de la aplicación, se han diseñado cuatro colecciones que incluyen documentos iguales. Esta decisión se ha tomado con el fin de facilitar la implementación de registros. Por otro lado, cabe destacar que, aunque no es una base de datos relacional, existen relaciones de claves dentro de documentos ya que, en algunos casos que comentaremos más adelante, es necesario mantener una relación con el fin de no tener información redundante en todas las colecciones. Las colecciones diseñadas para el desarrollo de esta aplicación son las siguientes:

- **User.** Esta colección complementa a la información guardada en *Firebase Authentication* relativa al usuario de la aplicación. Se genera con la misma clave que *Firebase Authentication* con el fin de realizar búsquedas más sencillas. Otra relación existente es la que hay entre el usuario y el comercio, que se realiza cuando el usuario tiene rol empleado.

```
{
  "name": String,
  "surname1": String,
  "surname2": String,
  "address": [String],
  "basket": [{
    "quantity": number,
    "articleId": String
  }],
  "email": String,
  "phone": String,
  "role": String,
  "shopId": String,
  "isBloqued": boolean,
  "isRemoved": boolean
}
```

**Figura 8. Ejemplo JSON documento User.**

- **Shop.** Esta colección incluye toda la información relativa a los comercios y empresas registradas en la aplicación. La principal relación existente en esta colección es la del usuario para identificar al administrador.

```
{
  "userId": String,
  "name": String,
  "email": String,
  "phone": String,
  "description": String,
  "address": String,
  "shippingMethods": [{
    "company": String,
    "price": number
  }],
  "isBlocked": boolean,
  "isRemoved": boolean
}
```

**Figura 9. Ejemplo JSON documento Shop.**

- **Order.** En esta colección se incluyen todos los pedidos realizados por un cliente a un comercio u empresa. Se guarda la relación entre el usuario y el comercio, así como una lista con las claves de los artículos existentes en el pedido.

```
{
  "address": String,
  "shippingMethods": [{
    "company": String,
    "price": number
  }],
  "items": [{
    "articleId": String,
    "quantity": number
  }],
  "shopId": String,
  "userId": String,
  "status": String,
  "Price": number
}
```

**Figura 10. Ejemplo JSON documento Order**

- **Article.** Esta colección contiene todos los artículos de los comercios de la aplicación. Se ha decidido separar los artículos a una colección a parte para que sea más fácil el tratamiento de estos, aunque sea necesario incluir una relación entre el comercio y este.



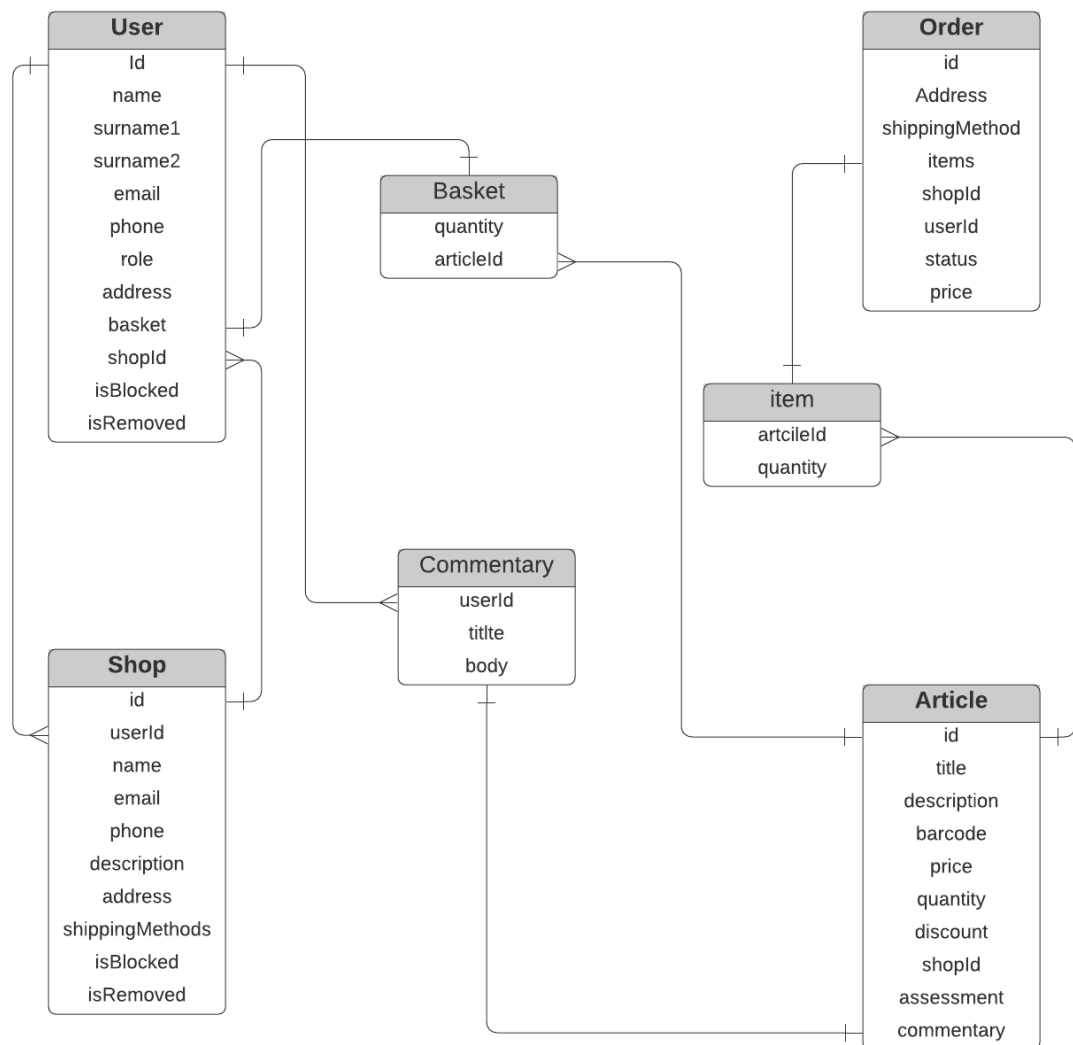
```

{"assessment": [number],
 "barcode": String,
 "commentary": [{
   "userId": String,
   "tittle": String,
   "body": String}]
 "description": String,
 "quantity": number,
 "price": number,
 "shopId": String,
 "title": String,
 "discount": number}

```

**Figura 11. Ejemplo JSON documento Article.**

Con el fin de facilitar la comprensión del modelo de datos y sus relaciones, en la figura 12 se puede ver un diagrama de entidad-relación.



**Figura 12. Esquema modelo de datos.**

En cuanto al gráfico, hay que destacar que las entidades *commentary*, *ítem* y *basket*, no tienen id ya que son objetos JSON dentro de los documentos que relacionan, pero sí tienen referencias a otros documentos. Por otro lado, *address* y *shippingMethods* también son objetos JSON iguales que los anteriores y, debido a la difícil compresión no han sido incluidos en el gráfico. Para finalizar, hay que destacar que en esta primera versión del proyecto se ha decidido que *address* sea de tipo *String* y que en la siguiente versión se incluya que la dirección sea un objeto JSON con los distintos parámetros relacionados con las direcciones (calle, número, ciudad ...).

## 5 Planificación

---

Como se indicó en el apartado 1.2 de los Objetivos de este proyecto, este debe realizarse mediante iteraciones y al finalizar cada una de estas se realizará una demo al cliente. En este apartado se va a hacer un desglose de la funcionalidad que se va a desarrollar en cada iteración de tres semanas durante el desarrollo de este TFG con el fin de obtener un producto básico de funcionalidad. La planificación completa se encuentra en el Anexo C.

### 5.1 Iteración 0: Análisis y diseño de la aplicación

Esta primera iteración, considerada como iteración 0, tuvo una duración de 4 semanas (16/09/2019-11/10/2019). En esta iteración no se realizó ningún tipo de desarrollo, ya que se centró en el análisis y diseño de la aplicación, además del estudio de las tecnologías que se iban a utilizar en el desarrollo de este proyecto.

### 5.2 Iteración 1: Superusuario

Esta iteración (14/10/2019-3/11/2019) se centró en las tareas relacionadas con el *superusuario*:

- Creación de la vista de usuarios.
- Creación del documento *user*.
- Creación de la vista de comercios.
- Creación del documento *shop*.
- Inclusión de los botones de acciones en las vistas.

### 5.3 Iteración 2: Gestión de usuarios

Esta segunda iteración (4/11/2019-24/11/2019), se centró en la gestión de los usuarios, como son el registro y autenticación:

- Creación de vista principal de la aplicación.
- Creación de formulario de registro.
- Creación de la lógica del registro.
- Creación del formulario de autenticado.
- Creación de la lógica de autenticado.
- Creación de la acción de cerrar sesión (vista).
- Creación de la lógica de cierre de sesión.
- Creación de las reglas de seguridad de Firebase.
- Creación de la vista del perfil de usuario.
- Creación de la lógica de borrado de usuario.
- Creación de la lógica de bloqueo de usuario.
- Creación de lógica de modificado de usuarios.

### 5.4 Iteración 3: Gestión de comercios

En esta iteración (25/11/2019-15/12/2019) se llevó a cabo toda la gestión de comercios, entre las que destacan las siguientes tareas:

- Creación del formulario de registro.
- Creación de la lógica de registro.
- Creación de la vista de perfil de comercio.
- Creación de la vista del comercio.
- Creación de la lógica de borrado de comercio.
- Creación de la lógica de bloqueo de comercio.
- Creación de lógica de modificado de comercio.

## 5.5 Gestión de artículos

Esta iteración (16/12/2019-5/1/2020) se centró en todas las tareas relacionadas con los artículos:

- Creación del documento *article*.
- Creación de la vista del inventario del comercio.
- Creación de la vista del artículo.
- Creación de la vista principal con búsqueda.
- Creación de formulario de artículos.
- Creación de lógica del formulario.
- Creación de lógica de recuperación con filtro por comercio y palabra clave.
- Inclusión de botones de acción en las vistas.
- Creación de lógica de eliminado de artículos.
- Creación de lógica de modificado de artículos.

## 5.6 Iteración 5: Carrito de la compra y finalización pedidos

En esta última iteración (6/1/2020-26/1/2020) se realizó toda la lógica de incluir artículos al carrito y realizar un pedido de forma simulada.

- Creación del documento *article*.
- Modificación del documento *user* para incluir el carrito.
- Creación del componente *basket*.
- Creación de la lógica para añadir un producto al carrito de la compra.
- Creación de la lógica para modificar la cantidad de un artículo en el carrito.
- Creación de la lógica para eliminar un producto del carrito.
- Creación de función para simular la pasarela de pago.
- Creación de la lógica para realizar un pedido.
- Creación de la vista de los pedidos para el comercio.
- Creación de la vista de los pedidos para el usuario.
- Creación de los botones de modificación de los pedidos.

## 6 Desarrollo

---

En este apartado se observará el proceso de desarrollo seguido durante el proyecto. Además, se van a incluir algunas de las funcionalidades del sistema, de acuerdo con lo establecido en el diseño mostrado en el capítulo anterior.

### 6.1 Entorno de desarrollo

Para el desarrollo de una aplicación Nuxt.js el sistema operativo de la máquina es indistinto, pero en este caso se ha decidido usar Windows 10, ya que, aunque es un entorno algo desconocido en el grado para el desarrollo, en el mundo laboral, es lo más utilizado.

En cuanto al entorno de desarrollo utilizado ha sido IntelliJ Idea, en particular Webstorm, que es el IDE especializado en la parte cliente que nos proporciona IntelliJ Idea. Esta elección ha sido por gusto personal del autor de este TFG ya que, cualquier otro entorno de desarrollo, como Microsoft VS Code, hubiese sido también válido. Aunque IntelliJ Idea también incluye la gestión del repositorio Git, se ha utilizado otra herramienta distinta: GitKraken, ya que para el autor de este TFG le es mucho más cómoda. El repositorio Git se encuentra en GitHub.

Para la fase de pruebas y depuración, se ha utilizado el navegador Google Chrome en última versión, ya que esta nos da una terminal de las peticiones al *backend*, así como distintas resoluciones. Además, junto con la terminal, se ha incluido la herramienta vue-dev-tools, incluida en este navegador con el fin de depurar el código.

### 6.2 Inicialización de la aplicación

El primer paso para el desarrollo de la aplicación es la creación de esta y añadir los distintos paquetes, dependencias y plataformas que se desean para que esta se ejecute y pueda ser desplegada.

El primer paso para inicializar una aplicación Nuxt.js junto con Firebase, es la creación de un proyecto Nuxt.js; para ello, desde una terminal de comandos y dentro de la ruta donde se desea generar el proyecto, se ejecuta el siguiente comando:

```
$ npx create-nuxt-app <project-name>
```

Este comando nos abre un menú donde deberemos elegir las características de la aplicación. Lo más relevante de estas elecciones fue seleccionar Vuetify.js para la creación de la interfaz de usuario y el tipo de aplicación SPA.

A continuación, es necesario crear el proyecto Firebase. Esto se realiza en dos partes: por un lado, se crea el proyecto en la consola de Firebase, en el Anexo D se realiza una explicación más a fondo esto. Y, por otro lado, se vincula el proyecto Nuxt.js con el proyecto Firebase creado anteriormente. Para ello, en una terminal de comandos y dentro de la ruta del proyecto nuxt.js creado, se ejecutan los siguientes comandos:

```
$ firebase login  
$ firebase init
```

El primer comando nos autentica frente a Firebase, y el segundo, inicializa el proyecto, si existen proyectos ya creados, nos permite seleccionar un proyecto, y continuar con la configuración. Este comando, nos crea un módulo *functions* para la generación de las distintas funciones que se ejecutarán en el *backend* y una serie de archivos de configuración entre los que destacan *firebase.json*, encargado de la comunicación cliente-servidor y *firestore.rule*, que nos indica las reglas de seguridad del servidor.

Para finalizar, hay que destacar que Nuxt.js incluye ya las librerías oficiales con las que se ha desarrollado este proyecto, entre las que destacan Vuex.js, Vuetify.js y vue-router.js; pero para la comunicación con Firebase, es necesario la instalación del módulo *firebase-tools* y para la generación de aplicaciones PWA el módulo *pwa*. Para ello, desde una terminal de comandos y dentro de la ruta del proyecto, se ejecuta el siguiente comando:

```
$ npm install firebase-tools
$ npm install @nuxt/pwa
```

También es necesario incluir en el fichero *nuxt.config.js* un nuevo módulo *pwa* con distinta información para el correcto despliegue en un futuro de la aplicación. Esta información consiste en el nombre de la aplicación, la descripción o el icono.

## 6.3 Frontend

Como se ha comentado anteriormente, en esta aplicación el lado cliente posee más carga computacional que el lado servidor, por tanto, el desarrollo comenzó con el lado cliente haciendo uso de las llamadas a las APIs de Firebase. En este apartado se van a poder visualizar las técnicas utilizadas para implementar el *frontend* de la aplicación.

### 6.3.1 Components

Vue.js se basa, principalmente en los componentes, que son partes de código reutilizable por otros elementos de la aplicación. Estos están compuestos de tres partes: código HTML para crear la vista, código CSS para dar diseño a la vista y código Javascript para incluir lógica al componente. Otra característica relevante de los componentes es la anidación de estos y su posterior comunicación, ya sea padre-hijo o hijo-padre.

En el desarrollo de este proyecto, se han creado componentes pequeños con funcionalidad determinada que, posteriormente construyen una funcionalidad más grande. Pero, centrándonos en el propio componente creado, para la parte del diseño gráfico se ha utilizado la librería Vuetify.js, que nos proporciona extractos de código HTML y CSS y, por otro lado, la lógica de estos componentes ha sido la comunicación entre estos y con el *store* de Vuex.js. En el Anexo E se puede observar una lista con todos los componentes creados para la aplicación.

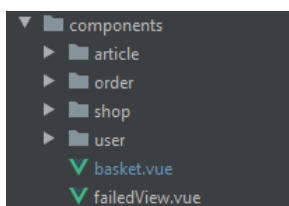


Figura 13. Components de la Aplicación

Para que los componentes puedan recibir y modificar los datos del *store*, durante la creación de estos deben “escuchar” la colección de datos con la que trabajará y, cuando este componente sea eliminado, (En el Anexo F, se explica el ciclo de vida de un componente Vue.js), dejar de “escuchar” esta colección. En el punto 5.3.4 entraremos más en detalle del funcionamiento del *store* de la aplicación.

```
mounted() {
  this.listenCol(Collection.User)
},
destroyed() {
  this.unlistenCol(Collection.User)
},
```

Figura 14. Ejemplo de escucha de colecciones.

Para tener acceso a las funciones del store (*getters*, *actions* y *mutations*), es necesario mapearlas dentro del componente mediante `...mapActions` o `...mapGetters` de la librería de Vuex.js. Como particularidad, cabe destacar que las acciones y mutaciones deben incluirse dentro de los métodos del *component* y los getters dentro del *computed*.

```
computed: {  
  ...mapGetters('dataset', ['getUser']),  
},  
methods: {  
  ...mapActions('dataset', ['listenCol', 'unlistenCol']),  
}
```

**Figura 15. Ejemplo de mapeo de las funciones del store dentro de un component.**

Una vez se hayan mapeado las funciones, estas se usan en todo el ámbito del componente como si fueran propias, consiguiendo así la reactividad de la aplicación.

La comunicación padre-hijo se realiza mediante propiedades del componente que se le pasan al declararlo; dentro de este, se utiliza como si fuese una variable propia declarándose las propiedades (*props*).

```
12 <article-card v-for="(item, i) in items()" :id="item.id" :key="i"></article-card>
```

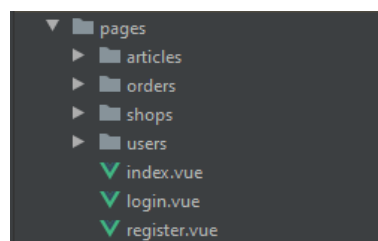
**Figura 16. Ejemplo de declaración de componente con propiedad.**

```
56 name: 'ArticleCard',  
57 props: ['id'],  
58 data: () => ({  
59   show: false,  
60 })),
```

**Figura 17. Ejemplo de declaración de props.**

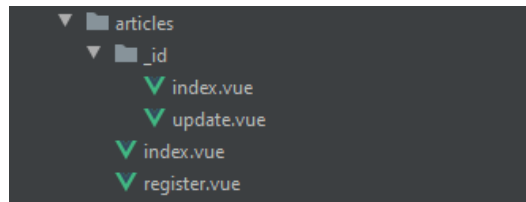
### 6.3.2 Pages

En Vue.js las *pages* (o páginas en español) son básicamente componentes, pero con diferencias sutiles. La principal diferencia es que las páginas son enrutadas y accedidas por parte del usuario final y el componente es instanciado por otro componente. Por otro lado, y como se ha comentado anteriormente, los componentes son partes de código que se reutiliza en distintos sitios de la aplicación, pero las páginas son únicas dentro de esta.



**Figura 18. Páginas de la aplicación.**

Con la ayuda de Nuxt.js, el enrutamiento de las páginas se realiza de forma automática con todas las páginas que existen dentro de la carpeta *pages* dentro del proyecto. Por tanto, como se puede ver en la imagen anterior, en la raíz de la aplicación tenemos *index.js*, que se referencia con `/`, además de *login.js* y *help.js* (que se identificará con `/login` y `/help`).

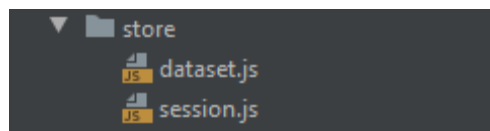


**Figura 19. Ejemplo de Páginas Anidadas.**

Cabe destacar que Nuxt.js nos permite tener páginas anidadas dinámicas; siguiendo el ejemplo de la imagen anterior, tenemos una página `/articles`, que contiene un `index`, donde se muestra la lista de artículos existentes en la aplicación, o los artículos relacionados con el comercio. Por otro lado, podemos acceder a una página individual por artículo indicando el `id` en el `path`, por ejemplo `/articles/{id}`. En este caso, existe una referencia a la página dentro del proyecto y dependiendo del identificador, se muestra una información u otra. En el Anexo G se hace una lista de las rutas de la aplicación.

### 6.3.3 Store

El `store` de Vuex.js nos permite tener los datos de la aplicación de forma reactiva; en este proyecto se han creado dos `store`: uno para gestionar las colecciones y documentos de la aplicación y otro para gestionar la sesión.



**Figura 20. Store existentes en el proyecto.**

Dentro de estos archivos existen 4 bloques para gestionar el estado:

- State.** El `state`, o estado en español, son los datos, en este caso, las colecciones o documentos existentes en Firebase, así como distintos datos estáticos que son necesarios para la aplicación.
- Getters.** Son las funciones que nos permiten obtener el estado dentro de los componentes. Normalmente, los `getters` aplican alguna regla o modificación del estado antes de ser devuelta al componente.
- Mutations.** Son funciones que aplican cambios en el estado.
- Actions.** Son acciones que se realizan en el estado, generalmente, contactando con el servidor y posteriormente se realiza una mutación para actualizar el estado.

```

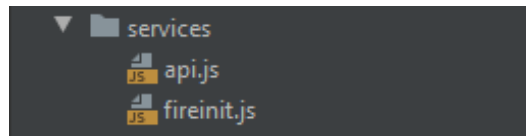
370 createModel ({ state, dispatch, commit }, { collection, data, id }) {
371   id !== null ? addDocumentById(collection, id, data) : addDocument(collection, data)
372 },
373 updateModel ({ state, dispatch, commit }, { collection, data, id }) {
374   return updateDocument(collection, id, data)
375 }
376 }

```

**Figura 21. Ejemplo de funciones Actions.**

Según la estructura seguida, se ha decidido que cuando exista una acción o mutación del estado de la aplicación, esta se aplique en Firebase y que esto provoque una actualización del estado mediante una mutación. Por tanto, las acciones llaman a servicios Firebase de forma transparente a partir de dos ficheros Javascript que son los encargados de la configuración y llamada.





**Figura 22. Ficheros Javascript para la conexión con Firebase.**

El archivo *fireinit.js*, contiene las funciones y configuraciones necesarias para la conexión con Firebase, además contiene los *endpoint* para la conexión de las *Firebase Functions* que se comentarán en el siguiente apartado del documento. Por otro lado, el archivo *api.js*, contiene funciones que llaman a la API de Firebase para el guardado, actualizado o eliminación de los documentos y colecciones de *Cloud Firestore*. Además, incluye una serie de constantes como son los Roles y las Colecciones con el fin de no repetir esta información por los componentes.

```
152 export const updateDocument = async function (collection, id, data){
153     const db = await getDB()
154     db.collection(collection).doc(id).set(data, {merge: true})
155 }
156
157 export const addDocumentById = async function (collection, id, data){
158     const db = await getDB()
159     db.collection(collection).doc(id).set(data)
160 }
```

**Figura 23. Ejemplo de funciones de api.js**

Cabe destacar que las funciones de la API de Firebase son funciones asíncronas, por lo que todas las funciones de *api.js* son asíncronas y las acciones que hagan uso de estas funciones deben esperar a obtener un resultado de esa llamada para continuar con su funcionamiento.

Para finalizar, indicar que el *store session.js* hace llamadas directas a la API de *Firebase Authentication* sin necesidad de pasar por las funciones anteriormente comentadas. Además, incluir que *session.js* y *dataset.js* se comunican entre sí con la ayuda de las funciones que nos facilita la librería *Vuex.js*.

## 6.4 Backend

El *backend* de este proyecto se realiza con una arquitectura *backendless* con la ayuda de Firebase, que contendrá la aplicación. En este apartado, se van a observar las principales características del *backendless* desarrollado.

### 6.4.1 Hosting

Una vez se inicializa el proyecto Firebase, es necesario indicar en qué *host* se desea desplegar la aplicación entre las distintas opciones que nos permite. En ese *host* se encontrará la aplicación y será donde se realicen las llamadas a los servicios de Firebase y donde se encontrarán las funciones que se ejecuten en el lado servidor. En este caso, se decidió desplegar en los servidores europeos.

Por otro lado, el hosting de Firebase nos permite, una vez desplegada la aplicación, incluir una URL propia siempre y cuando tengas el poder del dominio, además de las distintas URL que nos proporciona Firebase.

## 6.4.2 Reglas de seguridad

Como se comentó en el punto 4.3.2 se ha desarrollado un conjunto de reglas de seguridad Firebase, para controlar la robustez de la aplicación. Las reglas de Firebase permiten indicar que usuario puede leer o escribir en las colecciones de *Cloud Firestore*. En este caso, y como en la aplicación existen una serie de roles, estas reglas se aplicarán en función del rol que exista en la sesión de la aplicación:

- **Sin Autenticarse:** Se podrá leer la colección *Article* y *Shop* y no se podrá escribir en ninguna colección.
- **Rol Superusuario:** Tiene permisos totales de lectura y escritura sobre las colecciones de la aplicación.
- **Rol Administrador:** Tiene permisos de lectura y escritura sobre las colecciones *User*, *Shop*, *Article* y *Order*.
- **Rol Empleado:** Tiene los mismos permisos que administrador, excepto la escritura de *User*, que sólo podrá realizar modificaciones sobre su propio documento.
- **Rol Usuario:** Tiene permisos de escritura sobre *Order* y *User*; y permisos de lectura para *Article* y *Shop*.

```
service cloud.firestore {
  match /databases/{database}/documents {
    function getRoleByUser() {
      return get(/databases/{database}/documents/user/
        $(request.auth.uid)).data.role
    }
    function itsMe () {
      return get(/databases/{database}/documents/user/
        $(request.auth.uid)).data.email == request.auth.token.email
    }
    function isSingIn() {
      return request.auth.uid != null
    }
    match /user/{document} {
      allow read, create: if true
      allow update, delete: if itsMe();
    }
    match /shop/{document} {
      allow read: if true
      allow create: if isSingIn() && (getRoleByUser() == "user" ||
        getRoleByUser() == "superadmin")
      allow update: if isSingIn() && (getRoleByUser() == "admin" ||
        getRoleByUser() == "superadmin")
      allow delete: if isSingIn() && (getRoleByUser() == "admin" ||
        getRoleByUser() == "superadmin")
    }
    match /order/{document} {
      allow read: if isSingIn()
      allow create: if isSingIn() && (getRoleByUser() == "user")
      allow update: if isSingIn() && (getRoleByUser() == "admin" ||
        getRoleByUser() == "employee" ||
        getRoleByUser() == "user")
      allow delete: if isSingIn() && (getRoleByUser() == "superadmin")
    }
    match /article/{document} {
      allow read: if true
      allow create: if isSingIn() && (getRoleByUser() == "admin" ||
        getRoleByUser() == "employee")
      allow update: if isSingIn() && (getRoleByUser() == "admin" ||
        getRoleByUser() == "employee")
      allow delete: if isSingIn() && (getRoleByUser() == "admin" ||
        getRoleByUser() == "employee" )
    }
  }
}
```

Figura 24. Reglas de Seguridad Implementadas.

### 6.4.3 Firebase functions

Las *Firebase Functions* son funciones que se ejecutan en la parte del servidor dentro de la nube y a las que son llamadas desde el cliente. Es lo más parecido a un servidor tradicional que se ha implementado en la aplicación.

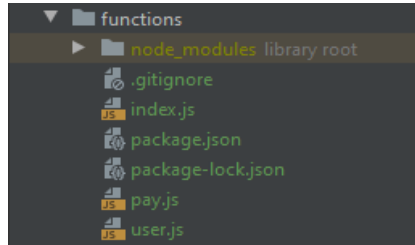


Figura 25. Proyecto de Firebase Functions.

Como se puede observar en la figura 25, existe un archivo *index.js* que redirige a los módulos *pay.js* y *user.js*, que es donde realmente se encuentran las funciones que se ejecutan en el servidor. En este apartado se han creado funciones en el servidor para simular la llamada a la pasarela de pago que, cuando se integre en realidad, será la salida a esta; y llamadas servidor a *Firebase Authentication* con el fin de obtener más funcionalidad que el API no ofrece.

```
exports.pay = functions.https.onCall( handler: (data, context) => {
  if (!context.auth) {
    throw new functions.https.HttpsError( code: 'failed-precondition', message: 'The function must be called ' +
      'while authenticated.' );
  }
  return pay.pay(data.orderId)
});
exports.pay = function (id) {
  const random = Math.random();
  const status = random < 0.9;
  return {id: status ? id.substr( from: 15, length: 5 ) : '', status: status}
}
```

Figura 26. Ejemplo de Función de Firebase.

Del mismo modo que la conexión del *store* con Firebase, las llamadas a Firebase no se realizan directamente en estos ficheros, si no que la implementación se realiza sobre el archivo *fireinit.js*, que contiene los *endpoints* para llamar a estas funciones que sí se ejecutan en el lado del servidor.

## 6.5 Despliegue

El despliegue de esta aplicación consta de 2 fases: en la primera se generará la aplicación PWA, y en la segunda se desplegará la aplicación en el *host* de Firebase.

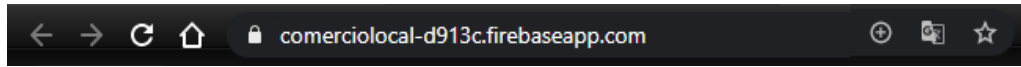
En cuanto a la generación de la aplicación PWA, desde una consola de comandos y en la ruta del proyecto, se ejecuta el siguiente comando:

```
$ nuxt generate
```

Una vez realizado, ya podemos desplegar la aplicación en Firebase, que se realiza, de la misma forma que antes, ejecutando en una terminal de comandos en la ruta donde se encuentra el proyecto:

```
$ firebase deploy
```

Una vez finalizado el despliegue en Firebase de la aplicación, este nos indica una URL, que si accedemos a ella nos muestra la aplicación. Vemos en el navegador que estamos accediendo al *host* de Firebase y que, además, se observa al lado de esta, la opción del “+” para poder descargar la aplicación tal y como se muestra en la figura 27.



**Figura 27. Resultado del despliegue.**

Por otro lado, también se puede navegar por toda la aplicación con la interfaz gráfica realiza en la URL: <https://comerciolocal-d913c.web.app/>. Este resultado se encuentra en el Anexo H de este documento.

## 7 Integración, pruebas y resultados

---

En este capítulo se va a mostrar la integración realizada del desarrollador con el cliente, así como todas las pruebas realizadas durante el desarrollo, incluyendo las demos con el cliente y sus resultados.

### 7.1 Integración

La aplicación, como se ha comentado en el capítulo 6, tiene una serie de relaciones con proveedores externos que han necesitado una integración.

En primer lugar, hay que destacar la integración de la aplicación con Firebase, en concreto con *Cloud Firestore* y *Firebase Authentication*. Estas herramientas son las mismas que se utilizarán una vez finalice el proyecto, pero para esta fase se han generado una serie de usuarios, comercios y artículos aleatorios e introducidos para la realización de todas las pruebas y, una vez finalice toda la fase de desarrollo y pruebas, estos datos serán eliminados. Por otro lado, la integración con *Firebase Authentication* no ha supuesto ninguna complicación, ya que se ha seguido toda la documentación del API para la conexión del cliente con este servicio.

Por otro lado, y aunque esto no ha sido desarrollado en las iteraciones realizadas durante el transcurso de este TFG, la aplicación deberá integrarse con una pasarela de pago externa por definir y con un API para la conexión con las agencias de envío que los comercios deseen incluir. En cuanto a la pasarela de pago, se ha realizado un *endpoint* mediante *Firebase Functions* que simula esta llamada a la pasarela.

Para finalizar, hay que destacar que las pruebas realizadas se han llevado a cabo en distintos navegadores, dispositivos y sistemas operativos para comprobar la correcta integración de la aplicación sobre estos y, gracias a la arquitectura *backendless* y al uso de librerías para las interfaces de usuario, no se han obtenido problemas de compatibilidad con ningún entorno.

### 7.2 Pruebas

En cuanto a las pruebas, hay que destacar que todas ellas se han realizado de forma manual sin automatización por medio de herramientas debido al poco tiempo para la realización de este proyecto y el alto coste de aprendizaje de estas herramientas. Estas pruebas se han realizado en un entorno local con la ayuda del navegador Google Chrome, su terminal y el *plugin* vue-dev-tools. Además, ha servido de apoyo la consola Firebase para comprobar el estado del *backend* en cada momento.

En primer lugar, se han realizado pruebas unitarias en la que se han ido probando los distintos componentes desarrollados durante las iteraciones que, posteriormente, comprenderán una funcionalidad completa dentro de la iteración y del proyecto. De estas pruebas se buscan resultados del comportamiento del componente de forma individual y de su correcta funcionalidad.

Por otro lado, las pruebas de integración se han centrado en pruebas completas de una funcionalidad. Frecuentemente, se realizaban a la finalización de una iteración, ya que estas se han dividido por funcionalidad. De estas pruebas se busca el correcto funcionamiento de la integración de los componentes y páginas que forman una funcionalidad concreta. Además, estas pruebas se han centrado en comprobar que los requisitos acordados se cumplían.

Cuando se realizaba una funcionalidad completa, se realizaban pruebas exhaustivas a esta y, posteriormente, pruebas regresivas de toda la aplicación para comprobar el correcto funcionamiento.

## 7.3 Demos continuas

Como se ha comentado anteriormente, tras la finalización de una integración se ha realizado una demo con el cliente enseñando los avances y obtener sus impresiones sobre lo desarrollado y las posibles modificaciones que fueran necesarias. Las fechas de las demos planeadas se encuentran en el anexo D donde se muestra la planificación.

En total, durante el desarrollo de este TFG se han realizado 5 demos en las que se presentó al cliente todo lo realizado durante esa iteración. De lo obtenido durante estas demos, que se tratará en el apartado siguiente, se acordó con el cliente abordarlo en iteraciones siguientes en función de la prioridad que el cliente marcará en función de su necesidad, tiempo y esfuerzo restante para la finalización de este TFG.

En cuanto a la funcionalidad entregada en cada demo, hay que destacar que tiene relación sobre lo desarrollado en cada iteración y puede observarse en el capítulo 5 de este documento, incluyéndose, como en la demo de la iteración 4, que se entregó funcionalidad relativa a iteraciones anteriores debido a los resultados obtenidos. Cabe destacar que no existió funcionalidad sin entregar en estas demos, por lo que la planificación realizada se ha cumplido.

## 7.4 Resultados

Durante el desarrollo de las pruebas unitarias se observaron una serie de problemáticas debido a la inexperiencia del autor de este TFG en las tecnologías utilizadas. Los primeros problemas encontrados estaban relacionados con la interfaz gráfica y su deseo de realizar una aplicación *responsive* ya que, a pesar de utilizar Vuetify.js, algunas imágenes o textos no tenían el estilo deseado y fue necesario realizar modificaciones sobre ello. El siguiente resultado obtenido de estas pruebas fueron los distintos problemas de comunicación con los datos y el hecho de que la aplicación utilizase Vuex.js. En las primeras comunicaciones no se escuchaba la colección deseada dentro del componente, por lo que no se mostraban los datos, haciendo así, que se visualizaran campos o listas vacías. Además, y debido a que las funciones de conexión con Firebase son asíncronas, cuando se realizaban modificaciones en el estado de la aplicación se observaban que estos cambios no se producían y era necesario esperar a que finalizara una acción para continuar con otra.

Por otro lado, los resultados obtenidos de las pruebas de integración, centradas más en pruebas de flujo completo, se observaron distintos problemas; el más reseñable fue en enrutamiento y la pérdida de datos una vez se pasaba de una página a otra, esto fue provocado por utilizar un tipo de llamada a una página que recargaba toda la aplicación, por lo que el estado se reiniciaba. Este problema se solventó investigando en la documentación de Nuxt.js y encontrando que existe otro tipo de enrutamiento donde se simula el cambio de página, pero el estado de la aplicación se mantiene.

Otro problema que reseñar fue la comunicación entre componentes vía propiedades o rutas dinámicas, se observó que los datos no existían debido a que no se trasladaban bien las propiedades. Del mismo modo que con el problema anterior, este problema fue debido a la inexperiencia del desarrollador con el framework y tras buscar información sobre esto, se obtuvo el resultado deseado.

En cuanto al resultado de las demos realizadas, como se ha comentado en el apartado 7.3, se han realizado un total de 5. En la primera demo, en la que se mostró la funcionalidad del superusuario, el reporte del cliente se centró en que esta era una funcionalidad bastante independiente del comercio completo, pero no encontraba problemas sobre ello y que, en futuras iteraciones, con más funcionalidad se observaría más su uso.

La segunda demo mostraba la funcionalidad del usuario: registro, inicio y cierre de sesión, así como la primera versión de la pantalla principal y su perfil. El resultado obtenido fue más del agrado del cliente, donde mostró una funcionalidad más usable para la aplicación final; pero el cliente mostró su desconformidad sobre el estilo de los formularios de logado y registro, así como del perfil, ya que no tenían incluidas las imágenes personalizadas. Otro problema que se obtuvo fue la validación de datos, ya que existían algunos campos sin esto. Estos problemas se solventaron en la siguiente iteración debido a que los componentes eran bastantes similares a los que se iban a desarrollar en esta.

La tercera demo se focalizó en la gestión de los comercios como son el registro y su modificación, y las vistas de estos, así como los problemas encontrados en la demo anterior y que se acordaron la implementación en esta iteración. Los resultados fueron bastantes parecidos a la demo anterior, indicando los problemas de interfaz gráfica anteriormente comentados debido a que usa componentes comunes.

En cuanto a la cuarta demo, focalizada en los artículos: incluir artículos al inventario, vistas, búsquedas y filtros; se focalizó en el desagrado del cliente con respecto a la vista del inventario de artículos por parte del administrador, observando que sobran algunos parámetros como era el id y la disminución de los caracteres de la descripción, incluso la eliminación de esta. Se acordó realizar estas mejoras en la siguiente iteración debido a su bajo coste de realización.

Para finalizar, la quinta demo se centró en unificar todo e incluir el carrito de la compra y una simulación de realizar pedidos. El resultado de esta demo fue satisfactorio, mostrando una primera versión de la aplicación funcional.

Debido a que el proyecto continúa en fase de desarrollo y no se ha producido una puesta en producción real, no se han obtenido datos reales de esta aplicación por parte del público ni cómo ha afectado a las ventas de los comercios que solicitaron la realización de la aplicación. Pero gracias a esta última iteración y mostrando un producto básico, se ha podido ver la primera fase del proyecto y el cliente está satisfecho con el trabajo realizado y ansioso de su finalización para poder comprobar cómo afecta a su negocio.





## 8 Conclusiones y trabajo futuro

---

### 8.1 Conclusiones

Tras el desarrollo de la aplicación y la obtención de los resultados correspondientes, es el momento de extraer todas las conclusiones pertinentes. Con el desarrollo realizado se ha conseguido implementar una aplicación PWA con la funcionalidad básica, pudiéndose mostrar lo que será el producto final con la ayuda del *framework* Nuxt.js. Esta aplicación final ayudará a los autónomos y pequeñas empresas de San Sebastián de los Reyes a fomentar sus negocios siendo un portal para ellos, además de poder vender sus productos dentro de esta aplicación.

El coste económico de lo realizado no se ha cuantificado actualmente, pero hay que destacar que, si el flujo tras la puesta en producción supera los 10GB de transferencia de datos al mes o 1 GB en datos almacenados, esto supondrá un coste aproximado de 20 a 30€ por los servicios ofrecidos en Google Firebase.

En cuanto al esfuerzo realizado ha sido alto debido a las tecnologías usadas y la dificultad del proyecto, ya que el *framework* y la arquitectura *backendless* no habían sido utilizadas antes y supuso un coste de aprendizaje, incluido al de desarrollo. Hay que destacar sobre este tema, que el esfuerzo y dificultad ha disminuido en función avanzaba el proyecto debido al aumento de conocimiento durante este periodo. Por otro lado, el tiempo utilizado para el desarrollo de este proyecto ha sido de aproximadamente 7 meses debido a que el autor de este TFG se encuentra trabajando a jornada completa en una empresa de desarrollo de software y ha supuesto un gran reto para este compaginar ambas tareas en el tiempo.

Otro aspecto que destacar es que se ha procurado realizar la aplicación lo más modular posible con el fin de realizar la integración de las posteriores iteraciones, así como con los posibles proveedores de la forma más sencilla, disminuyendo así el esfuerzo en futuras operaciones. Con respecto a esto, cabe indicar que también se ha procurado seguir un código lo más limpio posible con el fin de facilitar la mantenibilidad una vez el producto final sea puesto en producción.

Por otro lado, hay que destacar el agrado del cliente tras las demos realizadas, en especial tras la última, observando un producto cuidado, robusto y con buena base de lo que puede llegar a ser la aplicación final.

Por último, y no menos importante, quiero destacar que este proyecto ha supuesto un gran desafío ya que me he enfrentado a nuevas tecnologías como son Nuxt.js y sus distintas librerías, así como aprender nuevos servicios Firebase ya que, durante el grado se han enseñado algunos de ellos, pero no se ha visto el cómputo total. También me gustaría destacar la complejidad que ha supuesto compaginar el trabajo y la realización de este TFG, han sido momentos de agobio, cansancio y desesperación que han sido superados con el único objetivo de finalizar este proyecto. Además, querría comentar lo que ha supuesto enfrentarse a un proyecto completo con una parte *frontend*, ya que mi trabajo está centrado solo en la parte *backend* y gracias a este proyecto he visto la funcionalidad de los clientes reactivos y sus servicios, lo que me ayuda a comprender mejor el funcionamiento de las aplicaciones en su conjunto.

## 8.2 Trabajo futuro

Como se ha comentado en reiteradas ocasiones durante todo el documento, durante el desarrollo de este TFG se han implementado una serie de funcionalidades divididas en cinco iteraciones, pero aún queda más funcionalidad expuesta en el análisis de este documento y que no ha sido posible desarrollar hasta el momento:

- **Gestión de Pedidos.** Actualmente, los pedidos realizados se muestran en dos ventanas: una para el usuario que lo realiza y otra para el comercio al que se lo realiza, pero no existe más iteración sobre ellos. Es necesario abordar la funcionalidad de cancelación y modificación de estados de los pedidos tal y como lo recogen los requisitos **RF37 Y RF38**.
- **Valoración.** Tal y como queda reflejado en el análisis, los artículos deben poder ser valorados con una puntuación del 0 al 5. Esta funcionalidad servirá para mostrar a los clientes nuevos la valoración de otros usuarios sobre el producto y sobre el comercio lo vende. Además, permitirá realizar filtros y ordenaciones por valoración.
- **Comentarios.** De la misma forma que el apartado anterior, es necesario incluir comentarios en los artículos, con el fin de valorar de una forma más explícita los artículos.
- **Gestión de Empleados.** Actualmente, solo los usuarios con rol Administrador pueden realizar funciones dentro de la aplicación con el comercio (incluir artículos, modificar artículos, visualizar pedidos...) pero tal y como comentó el cliente y queda reflejado en el análisis de requisitos, es necesario la existencia de un rol Empleado para que otros miembros del comercio puedan realizar algunas funciones. Es necesario implementar el registro de los empleados y aplicar la lógica necesaria para controlar el acceso de estos sobre la base de datos.
- **Integración con Imágenes.** Es necesario que los perfiles, los comercios y los artículos tengan imágenes propias y descriptivas. Para ello, debe utilizarse *Firebase Storage*, modificar los distintos formularios de registro para guardar las imágenes y actualizar los componentes que utilicen imágenes para utilizar las guardadas.
- **Inclusión Pasarela de Pago.** Actualmente se simula la llamada a una pasarela de pago; es necesario contactar con los distintos bancos o paypal para conectar la aplicación con una pasarela de pago externa y conseguir pagos reales.
- **Inclusión API Métodos de Envío.** Se desea que cada comercio pueda indicar el método de envío que desea utilizar para sus pedidos. Para ello, es necesario crear un API que conecte la aplicación con los distintos distribuidores y que cuando se realice un pedido, envíe automáticamente la solicitud de envío a la compañía correspondiente.

## Referencias

---

- [1] yeeply, «5 Tipos de desarrollo de aplicaciones web más relevantes,» Yeeply, 2012. [En línea]. Available: <https://www.yeeply.com/blog/6-tipos-desarrollo-de-aplicaciones-web/>.
- [2] «Vender en Amazon España,» Amazon Services, 2020. [En línea]. Available: [https://services.amazon.es/servicios/vender-por-internet/caracteristicas-y-ventajas.html?ref=ases\\_soa\\_fb\\_hnav](https://services.amazon.es/servicios/vender-por-internet/caracteristicas-y-ventajas.html?ref=ases_soa_fb_hnav).
- [3] «¿Qué son las Aplicaciones Web? Ventajas y Tipos de Desarrollo Web,» Wiboo, 2017. [En línea]. Available: <https://wiboomedia.com/que-son-las-aplicaciones-web-ventajas-y-tipos-de-desarrollo-web/>.
- [4] «Ionos,» Ionos by 1&1, 2020. [En línea]. Available: <https://www.ionos.es/>.
- [5] «Cuál es la utilidad de las aplicaciones cliente/servidor,» culturacion, [En línea]. Available: <https://culturacion.com/cual-es-la-utilidad-de-las-aplicaciones-clienteservidor/>.
- [6] «Amazon,» Wikipedia, 2020. [En línea]. Available: <https://es.wikipedia.org/wiki/Amazon>.
- [7] L. Alonso, «Cómo funciona Amazon Marketplace,» Marketin4Ecommerce, 9 Marzo 2016. [En línea]. Available: <https://marketing4ecommerce.net/amazon-mecanica-de-un-marketplace/>.
- [8] «1&1 Ionos,» Wikipedia, 2020. [En línea]. Available: [https://es.wikipedia.org/wiki/1%261\\_Ionos](https://es.wikipedia.org/wiki/1%261_Ionos).
- [9] M. A. Alvarez, «Qué es una SPA,» desarrolloweb, 29 noviembre 2016. [En línea]. Available: <https://desarrolloweb.com/articulos/que-es-una-spa.html>.
- [10] I. Ramírez, «¿Qué es una Aplicación Web Progresiva o PWA?,» Xataka, 3 julio 2018. [En línea]. Available: <https://www.xataka.com/basics/que-es-una-aplicacion-web-progresiva-o-pwa>.
- [11] D. Laballós, «¿Cómo funcionan las aplicaciones móviles?,» diegolaballos, 24 julio 2019. [En línea]. Available: <https://diegolaballos.com/blog/como-funcionan-las-aplicaciones-moviles/>.
- [12] «Introduction to the Angular Docs,» Angular.io, 2020. [En línea]. Available: <https://angular.io/docs>.
- [13] A. Basalo, «Angular CLI,» desarrolloweb, 14 junio 2016. [En línea]. Available: <https://desarrolloweb.com/articulos/angular-cli.html>.
- [14] «Empezando,» reactjs, 2020. [En línea]. Available: <https://es.reactjs.org/docs/getting-started.html>.
- [15] «What is Vuex?,» vuejs, 2020. [En línea]. Available: <https://vuex.vuejs.org/>.
- [16] D. Ortego Delgado, «Diferencias entre Vue.js y React,» OpenWebinars, 19 septiembre 2017. [En línea]. Available: <https://openwebinars.net/blog/diferencias-entre-vuejs-y-react/>.
- [17] «The Progressive,» nuxtjs, 2020. [En línea]. Available: <https://nuxtjs.org/>.
- [18] jdonsan, «VueJS: El ciclo de vida de un componente,» elabismodenuall, 5 mayo 2017. [En línea]. Available: <https://elabismodenuall.wordpress.com/2017/05/05/vuejs-el-ciclo-de-vida-de-un-componente/>.
- [19] «What is Vuex?,» vuejs, 2020. [En línea]. Available: <https://vuex.vuejs.org/>.
- [20] «What's the difference?,» vuetifyjs, 2020. [En línea]. Available: <https://vuetifyjs.com/en/introduction/why-vuetify/>.
- [21] «Base de datos relacional,» Wikipedia, 2020. [En línea]. Available: [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_relacional](https://es.wikipedia.org/wiki/Base_de_datos_relacional).
- [22] «SQL,» Wikipedia, 2020. [En línea]. Available: <https://es.wikipedia.org/wiki/SQL>.
- [23] «NoSQL,» Wikipedia, 2020. [En línea]. Available: <https://es.wikipedia.org/wiki/NoSQL>.
- [24] Javier, «NoSQL vs SQL; principales diferencias y cuándo elegir cada una de ellas,» Pandorafms, 18 noviembre 2015. [En línea]. Available: <https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/>.
- [25] «Qué es AWS,» AWS Amazon, 2020. [En línea]. Available: <https://aws.amazon.com/es/what-is-aws/>.
- [26] «Firebase,» Firebase Google, 2020. [En línea]. Available: <https://firebase.google.com>.
- [27] A. Alonso, «Backend as a Service o como prescindir de un backend developer,» ADRIANALONSO.ES, 12 abril 2017. [En línea]. Available: <https://adrianalonso.es/desarrollo-web/backend-service-o-como-prescindir-de-un-backend-developer/>.
- [28] L. Jimenez, «Comparativa de desarrollo de aplicaciones,» Rubrika, [En línea]. Available: <https://rubrika.es/marketing-digital/desarrollo-de-aplicaciones/comparativa-desarrollo-aplicaciones/>.



## Glosario

---

API	Application Programming Interface
MVP	Minimum Viable Product
TFG	Trabajo Fin de Grado
SQL	Structured Query Language
NoSQL	No Structed Query Language
PWA	Progressive Web Application
SPA	Single Page Application
AWS	Amazon Web Service
MVC	Model View Control
MVVC	Model View View Model
HTML	Hipertext Markup Language
CSS	Cascading Style Sheet
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
DOM	Document Object Model
UI	User Interface
IDE	Integrated Development Environment



## Anexos

---

### A Casos de uso

En este anexo se describen los casos de uso de la funcionalidad más relevante de la aplicación realizado en la fase de análisis de este proyecto.

Identificador	CU-01	Título	Login
Autor	David García García		
Fecha	17-09-2020		
Actores Involucrados	Cualquier usuario registrado en la aplicación		
Resumen	El usuario accede a la aplicación e introduce sus credenciales en el panel de inicio de sesión y, si estos son correctos, el inicio de sesión se realiza.		
Pre-Condiciones	Estar registrado en el sistema.		
Post-Condiciones	El usuario ha iniciado sesión y accede a las distintas funcionalidades que le ofrece el sistema en función de su rol.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Introducir correo electrónico y contraseña.		2. Comprobar credenciales. 3. Otorgar acceso al usuario. 4. Redirigir al usuario a la pantalla correspondiente.	
Caminos Alternativos			
1.b El usuario introduce incorrectamente el correo electrónico o la contraseña. 3.b El sistema rechaza el inicio de sesión. 5. Mostrar una alerta al usuario de credenciales incorrectas y mostrar posibilidad de intentar de nuevo.			
Páginas Involucradas	Página Principal de cada rol de usuario y login		

**Tabla 1. Caso de uso CU-01**

Identificador	CU-02	Título	Logout
Autor	David García García		
Fecha	17-09-2020		
Actores Involucrados	Cualquier usuario autenticado en la aplicación		
Resumen	El usuario autenticado en la aplicación, cuando desee finalizar de realizar acciones, se des autentica de la aplicación.		
Pre-Condiciones	Haber iniciado sesión en la aplicación.		
Post-Condiciones	El usuario se ha des autenticado de la aplicación y se encuentra en la página principal de la aplicación.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Pulsa el botón de logout en el menú desplegable de la aplicación		2. Comprobar estado de la sesión. 3. Cierra sesión de la aplicación 4. Redirigir al usuario a la pantalla principal de la aplicación.	
Caminos Alternativos			
Páginas Involucradas	Menú desplegable y página principal		

**Tabla 2. Caso de uso CU-02**



Identificador	CU-03	Título	Registrar comercio
Autor	David García García		
Fecha	17-09-2020		
Actores Involucrados	Usuario, Superusuario.		
Resumen	El usuario registrado y autenticado puede registrar un comercio si lo desee con el fin de dar a conocer su comercio y vender sus artículos.		
Pre-Condicion	Estar autenticado en la aplicación. Tener un comercio real.		
Post-Condicion	El comercio queda registrado en el sistema y vinculado al usuario que lo ha registrado como administrador de este.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Pulsar el botón de registrar comercio.  3. Rellenar los datos relevantes del comercio y de contacto.  7. Incluir las distinas imágenes que solicita la aplicación		2. Redirigir al formulario de registro de comercio.  4. Validar los datos introducidos. 5. Guardar en base de datos el comercio. 6. Modificar el usuario para cambiar el rol a Administrador.  8. Guardar las imágenes en base de datos. 9. Mostrar ventana principal de administrador.	
Caminos Alternativos			
3.b Rellenar algún campo incorrecto. 4.b Encontrar error en la validación de datos. 9.b. Mostrar alerta de error en el campo incorrecto y dejar al usuario modificar el formulario.			
Páginas Involucradas	Página principal, Formulario y Dashboard Administrador.		

**Tabla 3. Caso de uso CU-03**

Identificador	CU-04	Título	Registrar usuario
Autor	David García García		
Fecha	18-09-2020		
Actores Involucrados	Cualquier usuario con acceso a la aplicación, Superusuario.		
Resumen	Cualquier usuario que acceda a la aplicación puede registrarse en esta para tener disponible toda la funcionalidad que necesite de autenticación.		
Pre-Condiciones	Entrar en el registro de usuarios.		
Post-Condiciones	El usuario se ha registrado y se muestra la página principal de la aplicación.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Rellena el formulario con los datos de contacto correspondientes junto con el correo electrónico y una contraseña.          4. Incluir una imagen como foto de perfil.		2. Validación de los datos. 3. Guardar en base de datos el nuevo usuario.   5. Guardar la imagen en base de datos. 6. Redirigir al usuario a la pantalla principal de la aplicación.	
Caminos Alternativos			
1.b Rellenar algún campo incorrecto. 3.b Encontrar error en la validación de datos. 6.b. Mostrar alerta de error en el campo incorrecto y dejar al usuario modificar el formulario.			
Páginas Involucradas	Página principal y formulario de registro		

**Tabla 4. Caso de uso CU-04**

Identificador	CU-05	Título	Añadir artículo al inventario
Autor	David García García		
Fecha	18-09-2020		
Actores Involucrados	Administrador o Empleado		
Resumen	Guardar en el inventario de la aplicación asociada al comercio un artículo nuevo.		
Pre-Condiciones	Haber iniciado sesión en la aplicación. Haber seleccionado la opción de nuevo artículo del dashboard del comercio.		
Post-Condiciones	Se ha guardado el nuevo artículo en base de datos y se muestra en el inventario del comercio y en el comercio online.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Introducir los datos relacionados con el artículo.    4. Incluir la imagen del artículo.		2. Comprobar la validez de los datos. 3. Guardar en base de datos el artículo  5. Guardar en base de datos la imagen 6. Redirigir al inventario del comercio.	
Caminos Alternativos			
1.b Rellenar algún campo incorrecto. 3.b Encontrar error en la validación de datos. 6.b. Mostrar alerta de error en el campo incorrecto y dejar al usuario modificar el formulario.			
Páginas Involucradas	Formulario de registro de artículo, Inventario Comercio		

**Tabla 5. Caso de uso CU-05**

Identificador	CU-06	Título	Buscar un artículo
Autor	David García García		
Fecha	18-09-2020		
Actores Involucrados	Usuario		
Resumen	Realizar una búsqueda por nombre de un artículo.		
Pre-Condiciones	Estar dentro de la aplicación y en la página principal		
Post-Condiciones	El usuario ha encontrado el artículo y ha accedido a su página.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Introduce el nombre del artículo en el buscador		2. Obtiene todos los artículos existentes.	
		3. Aplica el filtro por nombre de los artículos.	
		4. Muestra los resultados obtenidos.	
5. Selecciona el artículo deseado.		6. Redirige a la página del artículo.	
Caminos Alternativos			
3.b Aplica el filtro, pero no hay resultados.			
4.b Se muestra un mensaje de que no existen resultados para la búsqueda.			
Páginas Involucradas	Página principal, Página del artículo		

**Tabla 6. Caso de uso CU-06**

Identificador	CU-07	Título	Añadir artículo al carrito
Autor	David García García		
Fecha	19-09-2020		
Actores Involucrados	Uusario		
Resumen	Un usuario desea incluir un artículo que le interesa al carrito de la compra para posteriormente realizar el pedido.		
Pre-Condiciones	Haber iniciado sesión en la aplicación. Haber encontrado el artículo deseado.		
Post-Condiciones	El artículo queda asociado al carrito del usuario.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Selecciona la cantidad que desea.  2. Pulsa el botón de añadir al carrito.		3. Se guarda el artículo en el carrito. 4. Se actualiza la base de datos del perfil del usuario.	
Caminos Alternativos			
0.b se pulsa el botón de añadir al carrito desde la miniatura del producto.			
Páginas Involucradas	Página principal, página del producto.		

**Tabla 7. Caso de uso CU-07**

Identificador	CU-08	Título	Realizar pedido
Autor	David García García		
Fecha	19-09-2020		
Actores Involucrados	Usuario		
Resumen	Se realiza un pedido para cada comercio con los artículos seleccionados por el usuario.		
Pre-Condiciones	Haber iniciado sesión en la aplicación. Tener todos los artículos queridos en el carrito de la compra. Tener la vista del carrito de la compra.		
Post-Condiciones	El pedido queda registrado en cada comercio y en el perfil del usuario.		
Curso Básico de Eventos			
Usuario		Sistema	
1. Pulsa el botón de pagar.		2. Comprueba las cantidades existentes son menores que la cantidad del artículo en el inventario	
		3. Crea un pedido por cada comercio que tenga artículos en el carrito.	
		4. Guarda los pedidos en base de datos.	
		5. Contacta con pasarela externa para realizar pedido.	
		6. Actualiza el pedido con el estado del pedido y el pago.	
7. Selecciona el método de envío.			
8. Introduce la dirección de envío.		9. Contacta con proveedores de envío.	
		10. Actualiza el pedido en base de datos.	
		11. Muestra mensaje de pedido finalizado.	
Caminos Alternativos			
2.b la cantidad que se desea comprar es mayor a la cantidad del inventario.			
11.b Se muestra mensaje de error.			
5c. Pago erroneo en la pasarela de pago o fallo comunicación.			
11.c Se muestra mensaje de error.			
9.d Fallo en la creación de la orden de envio o fallo de comunicación.			
11.d Se muestra mensaje de error.			
Páginas Involucradas	Menú desplegable y página principal		

**Tabla 8. Caso de uso CU-08**

## B Maquetas

En este anexo se van a mostrar el resto de las maquetas realizadas durante el diseño de la aplicación, que no coinciden con todas las maquetas del proyecto, si no con una muestra del estilo y funcionalidad deseada.

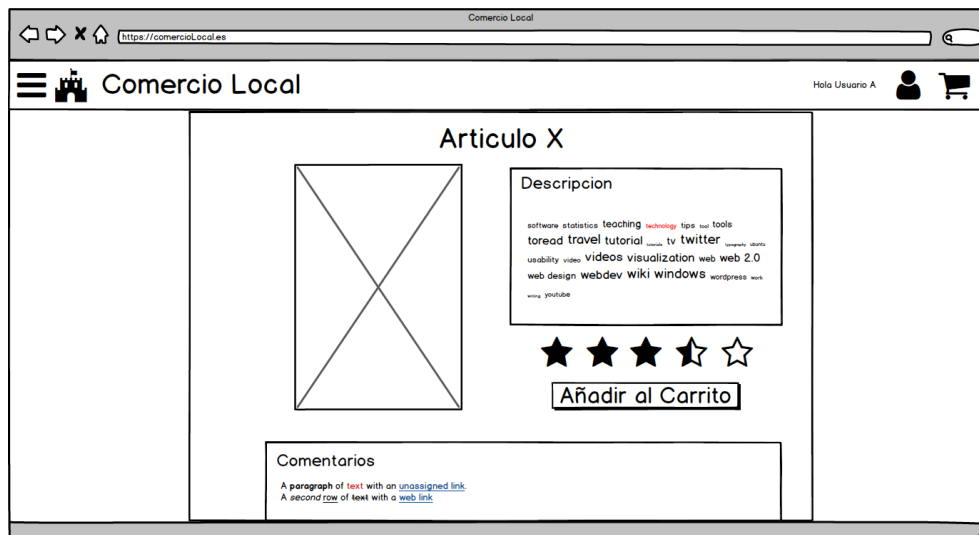


Figura 28. Maqueta de la vista artículos.

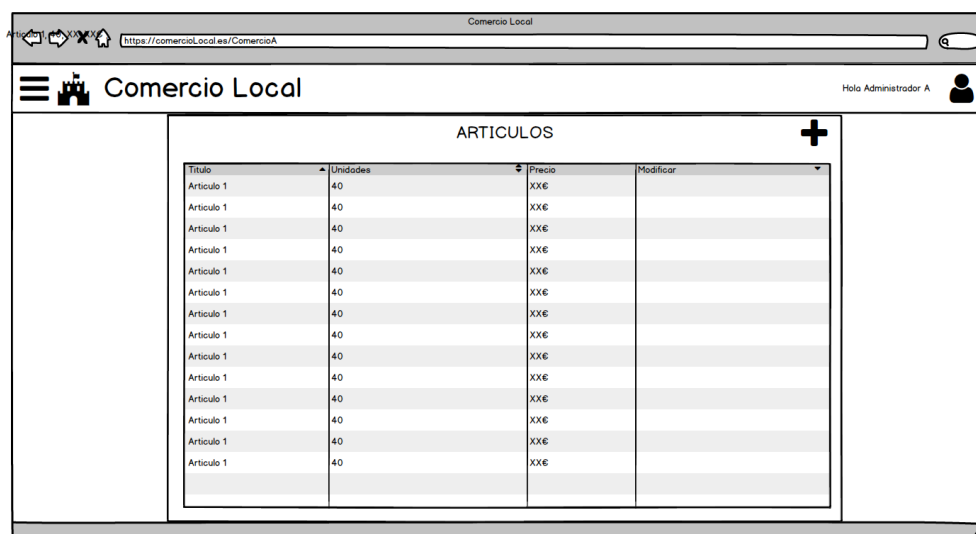


Figura 29. Maqueta del inventario artículos asociados al comercio

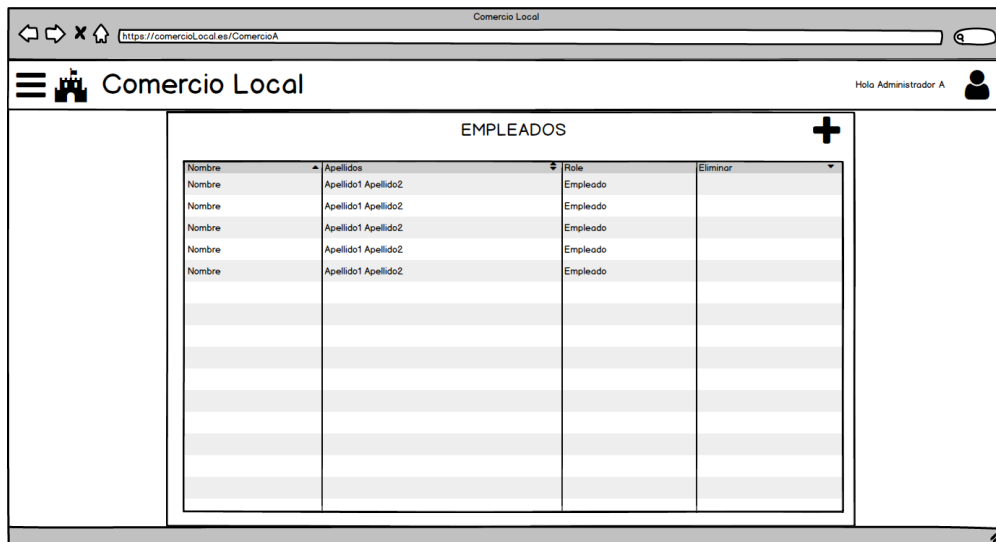


Figura 30. Maqueta del listado empleados comercio.

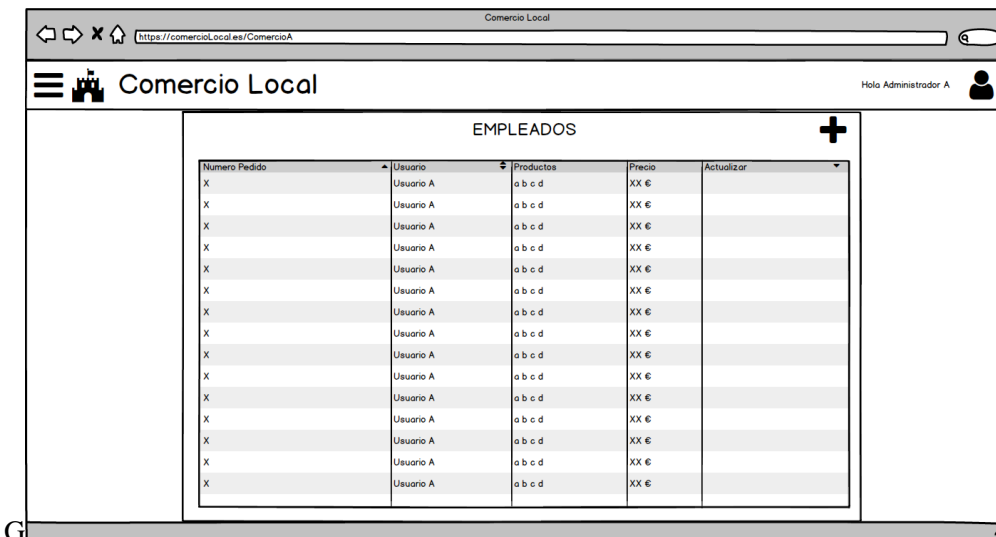


Figura 31. Maqueta de la lista pedidos asociados al comercio.

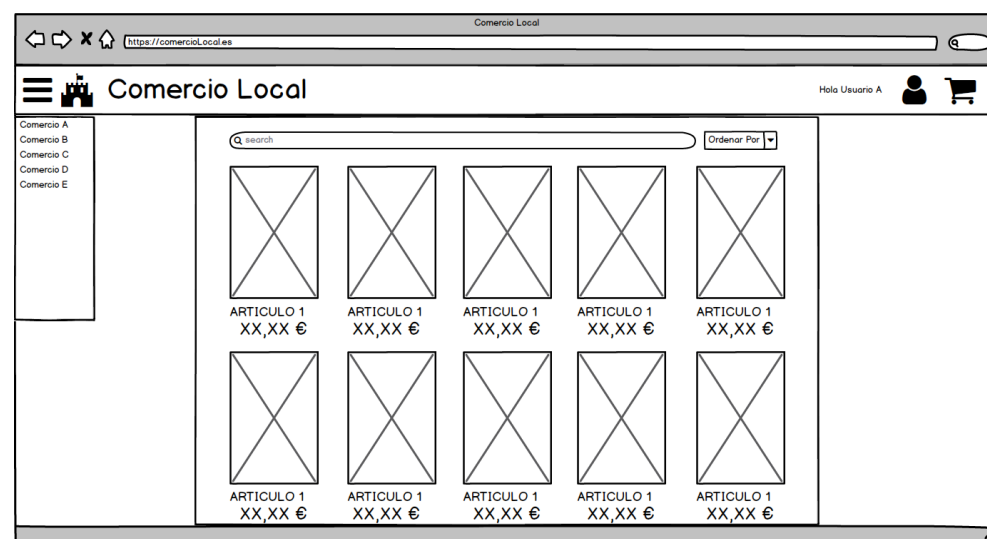
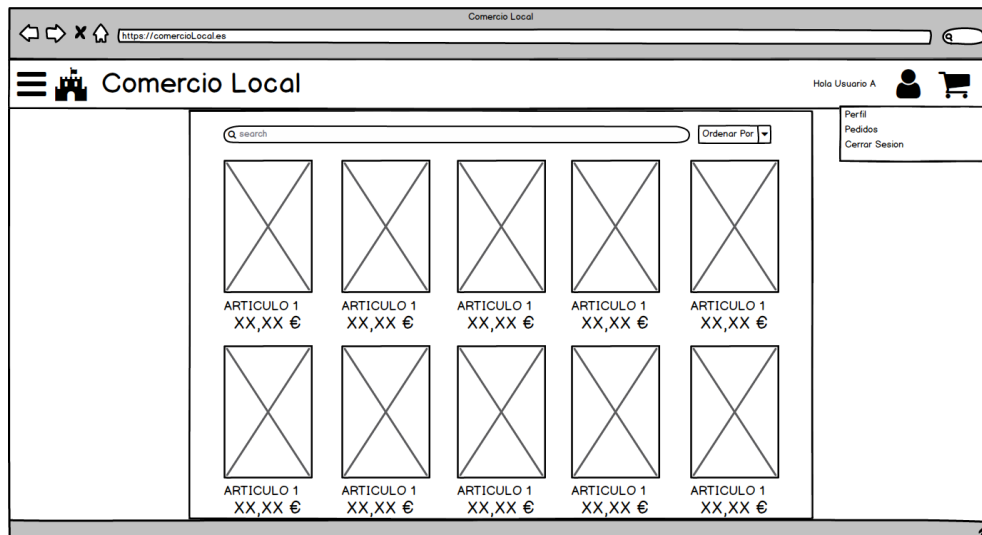
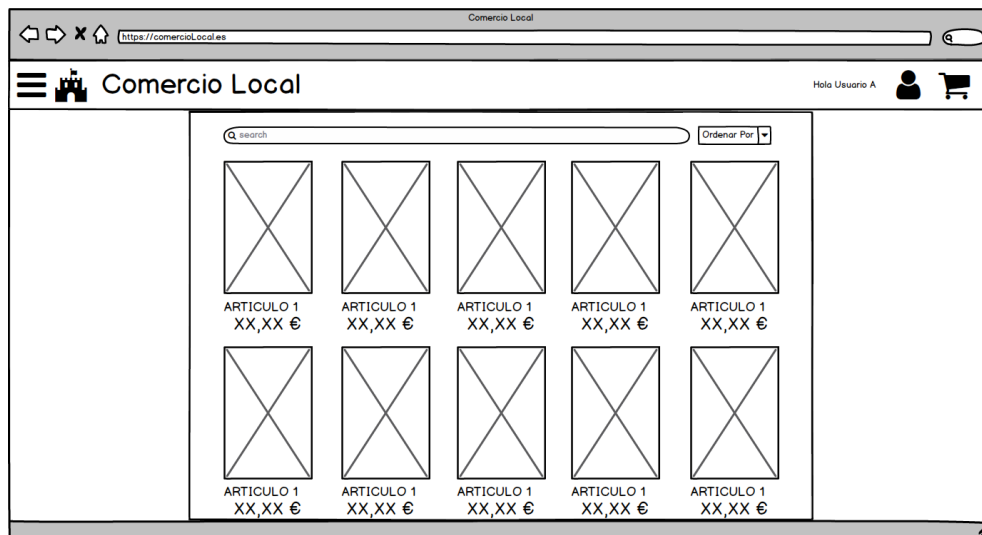


Figura 32. Maqueta de la vista principal con menú desplegable.

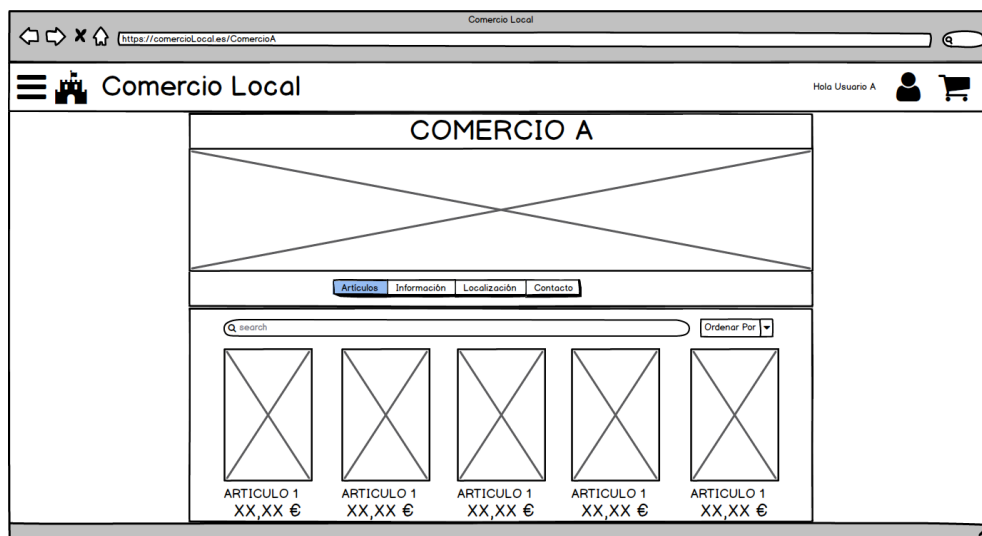




**Figura 33. Maqueta de la vista Principal con menú de usuario.**



**Figura 34. Maqueta de la vista principal usuario logado.**



**Figura 35. Maqueta de la vista principal comercio**



## C Planificación

Como se comenta en el capítulo 5, se realizan cinco iteraciones durante el transcurso de este proyecto. La planificación de trabajo se realiza teniendo en cuenta que, el autor de este TFG trabaja a tiempo completo en una empresa de desarrollo software y utiliza el tiempo libre para la realización de este proyecto. Se estima que cada semana este puede dedicar entre 16 y 20 horas al desarrollo del proyecto.

Se pone como límite para el desarrollo de las iteraciones incluidas en este TFG el mes de enero de 2020, con el fin de dedicar los meses de febrero y marzo del 2020 a la escritura de este documento y solucionar los distintos problemas que hayan surgido. En cuanto al inicio del desarrollo, este se realiza una vez finalizado el análisis y diseño comentado anteriormente como iteración 0, que tiene una duración real de 4 semanas comprendidas entre el mes de septiembre y octubre del 2019.

Como se puede observar en el calendario de la figura 36, existen un total de 15 semanas para realizar las cinco iteraciones que se desean incluir en el desarrollo, por lo que quedan 3 semanas por iteración. En cuanto a las tutorías realizadas, estas se realizaban el viernes de después de finalizar una iteración con el fin de preguntar las distintas dudas surgidas en el desarrollo de la iteración anterior, así como mostrar al tutor de este TFG el avance realizado sobre el proyecto. Por otro lado, las demos se realizaban el domingo de finalización de iteración.

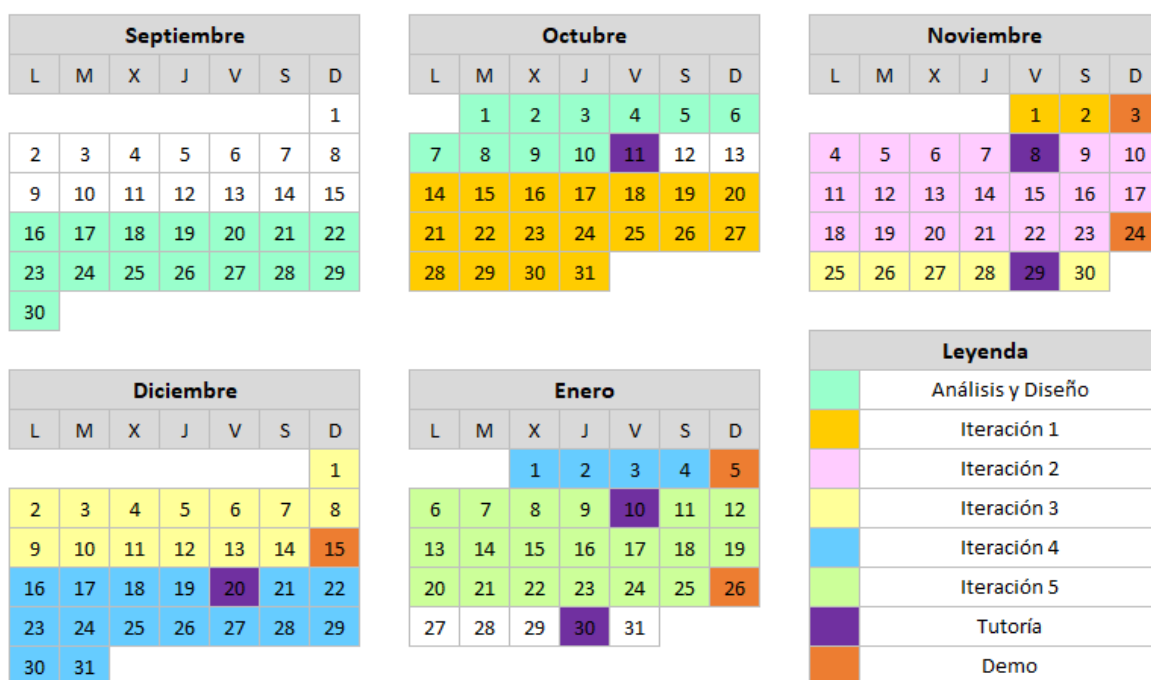
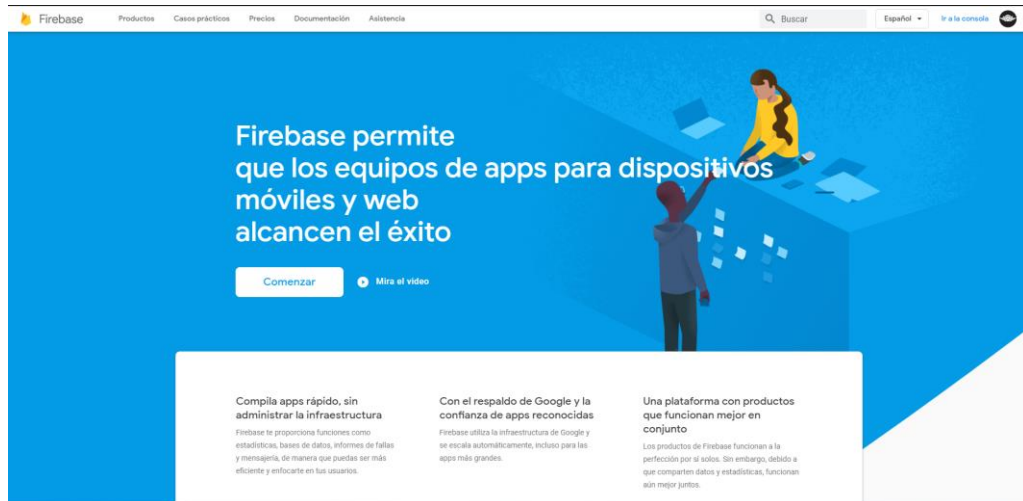


Figura 36. Calendario de la Planificación.



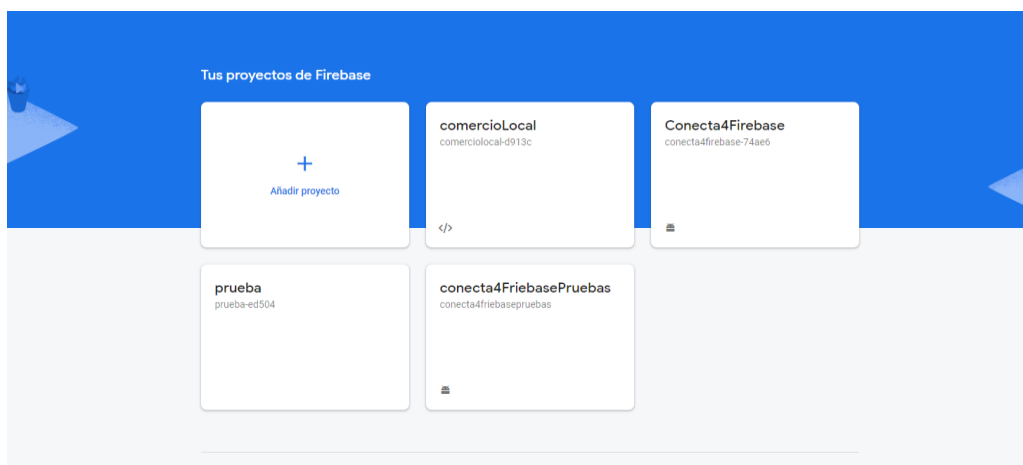
## D Consola Firebase

En este anexo se explica el funcionamiento de las distintas acciones que se pueden realizar mediante la consola Firebase. En primer lugar, para acceder a esta, es necesario tener una cuenta Google y acceder a la URL: <https://firebase.google.com/>.



**Figura 37. Página principal Firebase.**

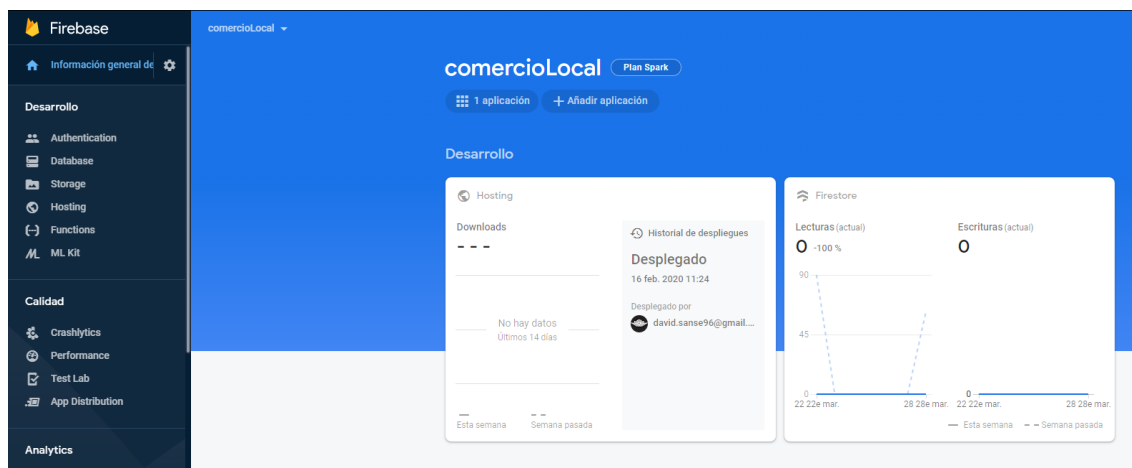
Como se puede observar en la figura 37, en la esquina superior derecha existe la opción ir a la consola. Pulsando en esta opción se abre una nueva ventana donde se muestran todos los proyectos existentes asociados a la cuenta de Google y la posibilidad de crear un nuevo proyecto (figura 38).



**Figura 38. Lista de proyectos Firebase.**

Si se selecciona la opción de “Añadir proyecto”, se inicia el proceso de creación de un proyecto mediante un formulario de 3 pasos entre los que se debe completar el nombre del proyecto, si se desea incluir Google Analytics o no al proyecto y seleccionar la ubicación de donde se encontrará el proyecto para obtener datos de Google Analytics. Cabe destacar que todo lo relacionado con Google Analytics es algo opcional y que, en cuanto al nombre del proyecto, este no debe estar ya seleccionado por otro usuario.

Una vez se genere este proyecto, se muestra la ventana desde donde gestionar este y desde donde se observan distintas gráficas de accesos a la aplicación una vez esta esté desplegada.



**Figura 39. Ventana principal proyecto Firebase.**













Accediendo a la opción de *Authentication*, se muestran todos los usuarios registrados en el sistema, desde donde se pueden incluir nuevos usuarios si se desea (figura 40), los métodos de inicio de sesión/registro (figura 41) y otras opciones relacionadas con la verificación del correo, o más gráficas analíticas relacionadas con los usuarios.

Desde este apartado se configura todo lo relacionado con *Firebase Authentication*. En primer lugar, una vez se cree el proyecto, se selecciona el tipo de inicio de sesión, para el desarrollo de este proyecto ha sido correo electrónico.

The screenshot shows the 'Authentication' section of the Firebase console, specifically the 'Users' tab. At the top, there are tabs for 'Users', 'Sign-in method', 'Templates', and 'Usage'. Below the tabs is a search bar with the placeholder text 'Buscar por dirección de correo electrónico, número de teléfono o UID de usuario' and an 'Añadir usuario' (Add user) button. The main content is a table of users.

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
hector@correo.com	✉	30 dic. 2019		4VLRW2wEYRCqoZ7u1fHmqKD...
susana@correo.com	✉	30 dic. 2019		BEp6Q5GVZ2fApQ9tokTjORdfnSP2
adrian@correo.com	✉	30 dic. 2019		H5w17P0UAKSKfaVPIL1Nycc1sC3
alba@correo.com	✉	30 dic. 2019		Q34SyjwoUlbhWHJ9IXq5Fsdn1DC2
victoria@correo.com	✉	30 dic. 2019		VtbGEyuyB3hfz4ZMVpq61496AYk1
maria@correo.com	✉	30 dic. 2019		X9A9qMyPTWRF070SiDaK3zRTpt...
jordi@correo.com	✉	30 dic. 2019	4 ene. 2020	Y7rxM9lCC0XtdAw0X8h0HVpk2p2

**Figura 40. Página de usuarios Firebase.**

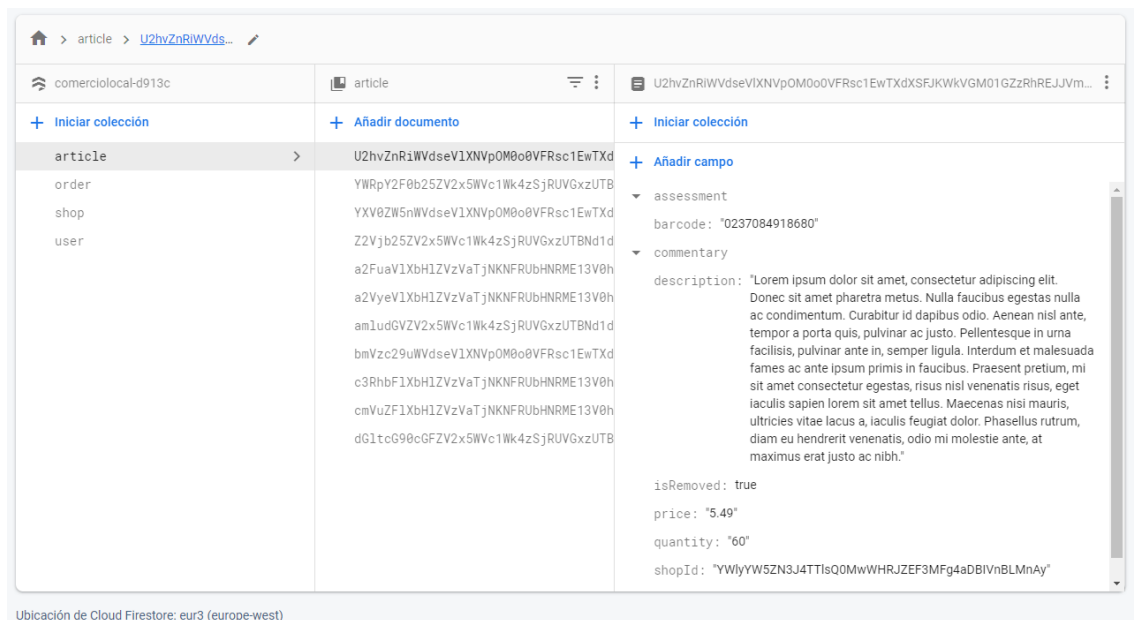
Proveedores de inicio de sesión	
Proveedor	Estado
 Correo electrónico/contraseña	Habilitada
 Teléfono	Inhabilitado
 Google	Inhabilitado
 Play Juegos	Inhabilitado
 Game Center <span>Beta</span>	Inhabilitado
 Facebook	Inhabilitado
 Twitter	Inhabilitado
 GitHub	Inhabilitado
 Yahoo	Inhabilitado
 Microsoft	Inhabilitado
 Apple	Inhabilitado
 Anónimo	Inhabilitado

**Figura 41. Página de tipos de inicio de sesión Firebase Authentication.**

Si accedemos a la opción de *Database*, accedemos a la base de datos de la aplicación. Existen dos opciones de base de datos: *Realtime Database* y *Cloud Firestore*. La segunda es la evolución de la primera y optimiza mejor el modelo de datos pudiendo realizar consultas más complejas y rápidas.

Cuando se crea el modelo de datos, es necesario seleccionar el *host* donde se encuentre la base de datos, en este caso, se seleccionó el *host* existente en Europa: *eur3* (europe-west)

Desde este apartado, se pueden observar todas las colecciones y documentos existentes en la aplicación, así como poder crear estos. También, desde este apartado se modifican las reglas de seguridad Firebase.

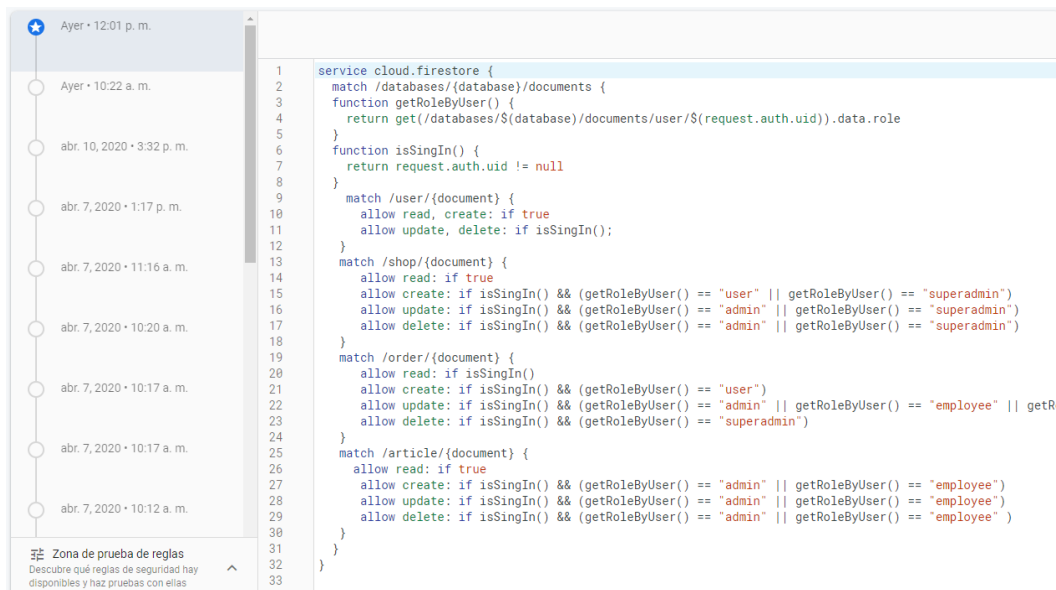


The screenshot shows the Firebase Cloud Firestore console. The breadcrumb navigation is 'home > article > U2hvZnRlWVds...'. The selected database is 'comercio-local-d913c'. The 'article' collection is selected, and a specific document is open. The document ID is 'U2hvZnRlWVdsV1XNVpOM0o0VFRsc1EwTXdXSfJKWkVGM01GZzRhREJVVm...'. The document contains the following fields:

- assessment**: (expanded)
- barcode**: "0237084918680"
- commentary**: (expanded)
- description**: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sit amet pharetra metus. Nulla faucibus egestas nulla ac condimentum. Curabitur id dapibus odio. Aenean nisl ante, tempor a porta quis, pulvinar ac justo. Pellentesque in urna facilisis, pulvinar ante in, semper ligula. Interdum et malesuada fames ac ante ipsum primis in faucibus. Praesent pretium, mi sit amet consectetur egestas, risus nisl venenatis risus, eget iaculis sapien lorem sit amet tellus. Maecenas nisi mauris, ultricies vitae lacus a, iaculis feugiat dolor. Phasellus rutrum, diam eu hendrerit venenatis, odio mi molestie ante, at maximus erat justo ac nibh."
- isRemoved**: true
- price**: "5.49"
- quantity**: "60"
- shopId**: "WlyYW5ZN3J4TTIsQ0MwWHRJZEF3MFg4aDBiVnBLMnAy"

At the bottom, it indicates the location of the Cloud Firestore: 'eur3 (europe-west)'.

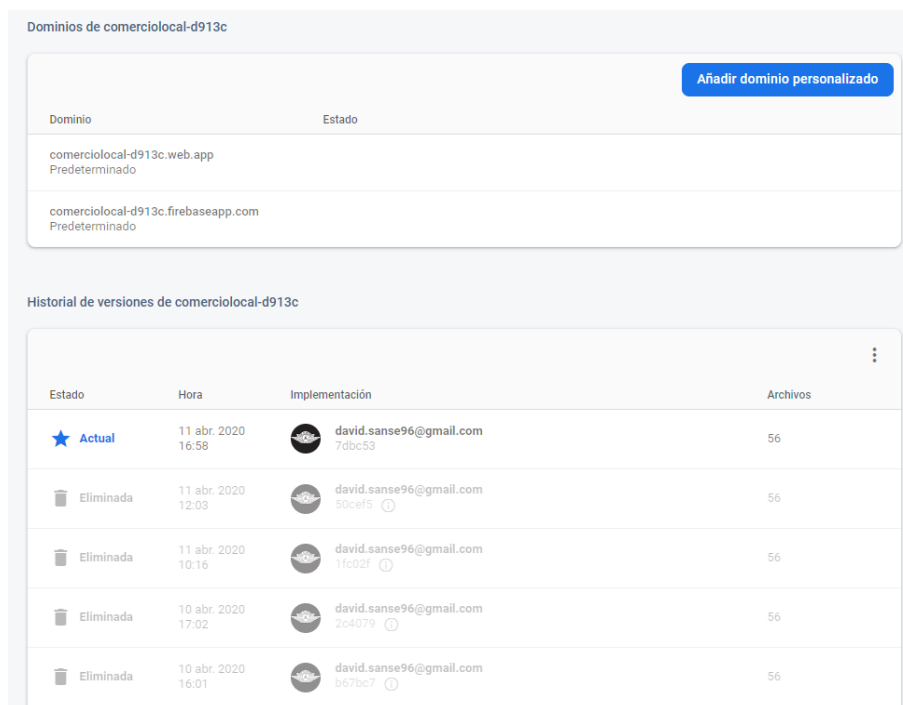
**Figura 42. Vista Cloud Firestore en consola Firebase.**



**Figura 43. Vista reglas seguridad Firebase.**

Por otro lado, en la pestaña de *Hosting*, cuando se genera, se incluye el lugar donde se encontrará este que, de la misma forma que anteriormente, se selecciona el servidor existente en Europa: *eur3* (europe-west). Además, indica una serie de pasos a seguir en el proyecto local, en nuestro caso, el proyecto Nuxt.js para inicializar el *host*, que consiste en instalar el módulo definido en el apartado 6.2.

Una vez el *host* sea inicializado, se muestra un panel de control donde se observan los distintos dominios asociados a la aplicación, así como la versión del proyecto desplegada actualmente y un historial de esta.



**Figura 44. Panel de control Hosting consola Firebase.**



En cuanto al apartado de *Functions*, al inicializar este nos indica que sigamos unos pasos similares a los de *Hosting*, para indicar el lugar del *host* donde se encontrarán las funciones, así como inicializar la aplicación con Firebase.

Una vez inicializado y desplegado el servidor, en estas ventanas se muestra una lista de las *Firebase Funcions* implementadas, así como la lista de registros (*logs*) del servidor para comprobar el funcionamiento de la aplicación.

Función	Activador	Región	Tiempo de ejecución	Memoria	Tiempo de espera
createUser	HTTP Solicitud https://us-central1-comerciocal-d913c.cloudfunctions.net/createUser	us-central1	Node.js 8	256 MB	60s
disabledUser	HTTP Solicitud https://us-central1-comerciocal-d913c.cloudfunctions.net/disabledUser	us-central1	Node.js 8	256 MB	60s
pay	HTTP Solicitud https://us-central1-comerciocal-d913c.cloudfunctions.net/pay	us-central1	Node.js 8	256 MB	60s

Registros de búsqueda...		Todo seleccionado	Todos los niveles de registro	
Hora	Nivel	Función y mensaje del evento		
11 abr. 2020				
4:57:30.106 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
4:57:57.884 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
4:58:04.353 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
4:58:31.051 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
5:01:11.569 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
5:01:11.569 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
5:01:11.594 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
5:01:11.736 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
5:01:11.736 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		
5:01:12.556 p...		<pre>{   "@type": "type.googleapis.com/google.cloud.audit.AuditLog",   "authenticationInfo": {     "principalEmail": "david.sanse96@gmail.com"   },   "requestMetadata": {     "callerI..."   },   "serviceName": "..." }</pre>		

**Figura 45. Panel Functions en consola.**

Para finalizar con la explicación de la consola de Firebase, hay que destacar que existe en todos los apartados explicados anteriormente un apartado donde se muestran los registros de acceso a la aplicación, así como en la página principal como se puede observar en la Figura 39. Esto nos ayuda a tener el control total sobre el estado de la aplicación una vez desplegada.



## E Componentes de la aplicación

En este anexo se encuentra el listado de todos los componentes de la aplicación. Con el fin de llevar una mejor estructura en el proyecto, se han creado subcarpetas dentro de la carpeta *components* del proyecto Nuxt.js para dividir estos en función de la página donde se usa.

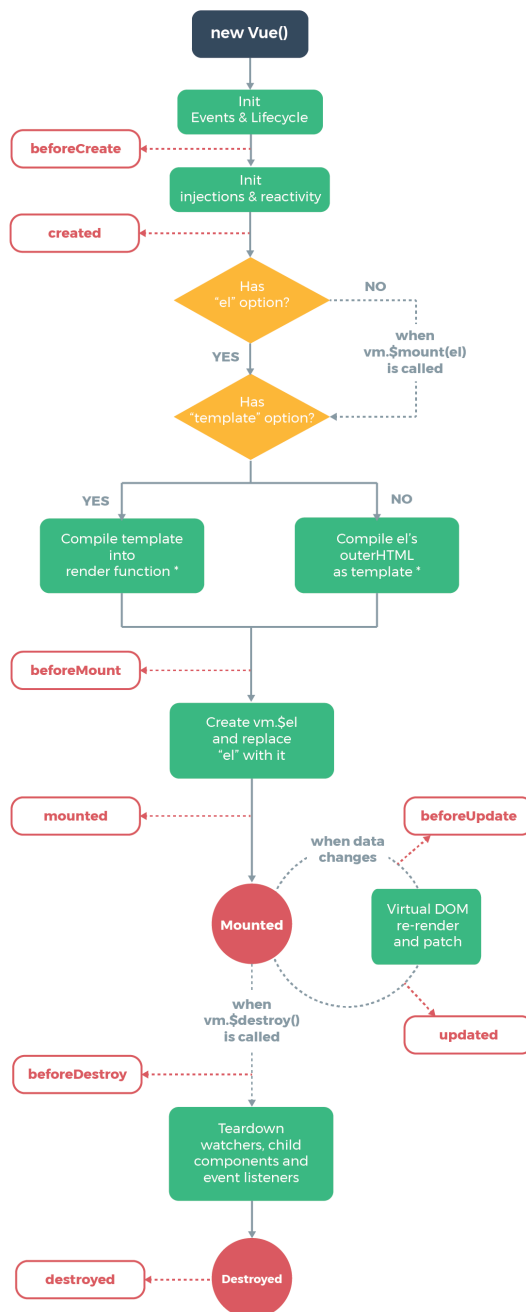
Nombre	Ruta en el Proyecto	Descripción
<i>basket.vue</i>	<i>/components/basket.vue</i>	Componente del carrito de la compra.
<i>failedView.vue</i>	<i>/components/failedView.vue</i>	Componente si existe fallo.
<i>articleCard.vue</i>	<i>/components/article/articleCard.vue</i>	Componente de la vista en miniatura de un artículo que se utiliza en index.
<i>articleForm.vue</i>	<i>/components/article/articleForm.vue</i>	Componente del formulario para registrar nuevos productos
<i>articleList.vue</i>	<i>/components/article/articleList.vue</i>	Componente de la lista de los productos para las vistas del administrador y superusuario.
<i>articleProfile.vue</i>	<i>/components/article/articleProfile.vue</i>	Componente de la vista principal de un artículo.
<i>articlesView.vue</i>	<i>/components/article/articlesView.vue</i>	Componente de la lista de artículos para los usuarios que utiliza las vistas en miniatura de los productos.
<i>articleUpdateForm.vue</i>	<i>/components/article/articleUpdateForm.vue</i>	Componente del formulario para modificar un artículo.
<i>orderList.vue</i>	<i>/components/order/orderList.vue</i>	Componente de la lista de los pedidos para las vistas del administrador, usuario y superusuario.
<i>shopContactInfo.vue</i>	<i>/components/shop/shopContactInfo.vue</i>	Componente con la información del contacto del comercio visible en el perfil del comercio.
<i>shopForm.vue</i>	<i>/components/shop/shopForm.vue</i>	Componente con el formulario para incluir un nuevo comercio.
<i>shopList.vue</i>	<i>/components/shop/shopList.vue</i>	Componente de la lista de comercios para la vista del superusuario.
<i>shopProfile.vue</i>	<i>/components/shop/shopProfile.vue</i>	Componente con el perfil del comercio.
<i>shopUserInfo.vue</i>	<i>/components/shop/shopUserInfo.vue</i>	Componente con la información del administrador del comercio visible en el perfil del comercio.
<i>shopUpdateForm.vue</i>	<i>/components/shop/shopUpdateForm.vue</i>	Componente con el formulario para modificar un comercio.
<i>shopView.vue</i>	<i>/components/shop/shopView.vue</i>	Componente que representa la vista del comercio para los usuarios.
<i>userForm.vue</i>	<i>/components/user/userForm.vue</i>	Componente con el formulario para registrar un usuario
<i>userList.vue</i>	<i>/components/user/userList.vue</i>	Componente de la lista de usuarios para la vista del superusuario.
<i>userProfile.vue</i>	<i>/components/user/userProfile.vue</i>	Componente del perfil del usuario.
<i>userUpdateForm.vue</i>	<i>/components/user/userUpdateForm.vue</i>	Componente con el formulario para modificar un usuario.

**Tabla 9. Lista de los componentes de la aplicación.**



## F Componente Vue.js

En este anexo se va a hacer una descripción más en profundidad del componente Vue.js y de su ciclo de vida. Como se ha comentado durante este documento, un componente en Vue.js es la unidad básica para el desarrollo de un proyecto; está compuesto por fragmentos de código HTML y CSS para generar la vista, y código Javascript para generar el modelo. Estos componentes son reutilizables por otros componentes o páginas, por lo que tienen un ciclo de vida dentro de la aplicación donde se generan y se cierran en cualquier momento.



\* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

Figura 46. Ciclo de vida componente vue.js.

Como se observa en la figura 46, el ciclo de vida de un componente tiene una serie de estados por los que pasa y que, nos da la posibilidad de realizar acciones antes y después de que el componente pase por ese estado. Para ello, Vue.js nos proporciona una serie de métodos que iremos viendo en función del estado al que hacen referencia. Estos métodos, además, nos ayudan a comprender el ciclo de vida del componente pudiendo incluir trazas en estos estados por los que pasa el componente Vue.js.

En la fase de la creación del componente, comprendida entre la instancia y el montaje en el DOM de este, Vue.js nos proporciona dos métodos y tienen la principal característica de que son los únicos que pueden interceptarse en el renderizado del servidor, ya que el resto, solo se pueden ejecutar en el navegador.

- ***beforeCreate***. Este método se ejecuta nada más instanciar el componente. En este momento no se han registrado aún los observadores de datos ni se han registrado eventos. Suele utilizarse para configurar parámetros internos u opciones propias del componente Vue.js como pueden ser librerías.
- ***created***. En este método ya se han registrado los observadores y métodos, por lo que ya existen los datos del estado, pero aún no ha sido renderizado ni incluido en el DOM. Suele utilizarse para iniciar las variables del estado de forma asíncrona. En el desarrollo de este proyecto, este método ha sido el usado para inicializar los escuchadores del *store* con el fin de obtener todos los datos necesarios para el uso del componente.

Una vez creado el componente, es el momento de iniciar la fase de montaje (renderización) de este y su futura inserción en el DOM. Existen dos métodos que se ejecutan durante esta fase:

- ***beforeMount***. Este método se ejecuta justo antes de insertar el componente en el DOM, no suele utilizarse para incluir funcionalidad extra, ya que se ejecuta justo después del método *created* y este suele contener toda la funcionalidad.
- ***mounted***. En este método se suelen realizar manipulaciones del componente en el DOM nada más inicializarlo como puede ser el inicializado de librerías externas.

Cuando el componente ya se encuentra registrado en el DOM este se actualiza de manera reactiva, por lo que entra en una fase de actualización antes de que este sea destruido. Durante esta fase, se renderiza y se actualiza el DOM en función de los cambios encontrados en el estado de la aplicación. De la misma forma que las anteriores fases, cuenta con dos métodos:

- ***beforeUpdate***. Este método se ejecuta nada más provocarse la actualización del estado, antes de comenzar la renderización nueva. Se suele utilizar para trazar los cambios que se van a incluir en el renderizado y realizar distintos cálculos en función de la unión de estados.
- ***updated***. Este método es similar al método *mounted*, ya que se ejecuta justo después de haber renderizado el componente con los cambios en el DOM.

Para finalizar, cuando el componente ya no es necesario para el usuario, este entra en la fase de destrucción, que consiste en eliminar su estancia de memoria y del DOM. Vue.js nos proporciona otros dos métodos relacionados con esta fase:

- ***beforeDestroy***. Este método se ejecuta justo antes de eliminar la instancia del DOM, sigue siendo un componente operativo y se puede acceder a todas sus propiedades. Suele utilizarse para eliminar eventos o escuchadores del componente.
- ***destroyed***. En este caso, suele utilizarse para limpiar los escuchadores de las colecciones del estado ya que, ha sido eliminado ya de todos los sitios el componente.

## G Páginas de la aplicación

Este anexo contiene todas las páginas implementadas durante el desarrollo del proyecto y que se encuentran en la carpeta *pages* del proyecto Nuxt.js tal y como se ha explicado en el apartado 6.3.2 de este documento.

Nombre Página	Ruta en el Proyecto	Descripción
Página principal	<i>pages/index.vue</i>	Página inicial de la aplicación.
Login	<i>pages/login.vue</i>	Formulario de inicio de sesión.
Ayuda	<i>pages/help.vue</i>	Página de ayuda de la aplicación.
Lista usuarios	<i>pages/users/index.vue</i>	Página para el superusuario visualice la lista de todos los usuarios de la aplicación.
Registro usuarios	<i>pages/users/register.vue</i>	Formulario para registrar usuarios en la aplicación.
Perfil usuario	<i>pages/users/_id/index.vue</i>	Perfil del usuario con el id indicado en la URL
Modificar usuario	<i>pages/users/_id/update.vue</i>	Formulario para modificar el perfil del usuario indicado en el path.
Lista comercios	<i>pages/shops/index.vue</i>	Página para que el superusuario visualice la lista de todos los comercios de la aplicación
Registro comercio	<i>pages/shops/register.vue</i>	Formulario para registrar un comercio
Página principal comercio	<i>pages/shops/_id/index.vue</i>	Página principal del comercio indicado en el path.
Modificar comercio	<i>pages/shops/_id/update.vue</i>	Formulario para modificar el comercio indicado en en path.
Lista pedidos	<i>pages/orders/index.vue</i>	Lista para que los usuarios y los administradores visualicen los pedidos que tienen asociados. El superusuario visualizará todos los pedidos existentes en la aplicación.
Lista artículos	<i>pages/articles/index.vue</i>	Lista para que los administradores visualicen los productos asociados a su comercio. El superusuario visualizará todos los productos existentes en la aplicación.
Añadir artículo	<i>pages/articles/register.vue</i>	Formulario para añadir un nuevo artículo al comercio asociado al administrador que realiza esta acción.
Vista Artículos	<i>pages/articles/_id/index.vue</i>	Vista del artículo indicado en en path.
Modificar artículo	<i>pages/articles/_id/update.vue</i>	Formulario para modificar el artículo indicado en el path.

**Tabla 10. Lista de páginas de la aplicación.**

En la tabla anterior se observan todas las páginas implementadas durante el proyecto y el lugar donde se encuentran dentro del proyecto para ayudar a la comprensión del enrutamiento dinámico que utiliza vue-router.js. Por otro lado, en la tabla 11, se muestran las rutas asociadas a estas páginas.

<b>Nombre Página</b>	<b>URL</b>	<b>Descripción</b>
Página principal	/	Página inicial de la aplicación.
Login	/login	Formulario de inicio de sesión.
Ayuda	/help	Página de ayuda de la aplicación.
Lista usuarios	/users	Página para el superusuario visualice la lista de todos los usuarios de la aplicación.
Registro usuarios	/users/register	Formulario para registrar usuarios en la aplicación.
Perfil usuario	/users/{id}	Perfil del usuario con el id indicado en la URL
Modificar usuario	/users/{id}/update	Formulario para modificar el perfil del usuario indicado en la URL
Lista comercios	/shops	Página para que el superusuario visualice la lista de todos los comercios de la aplicación
Registro comercio	/shops/register	Formulario para registrar un comercio
Página principal comercio	/shops/{id}	Página principal del comercio indicado en la URL
Modificar comercio	/shops/{id}/update	Formulario para modificar el comercio indicado en la URL
Lista pedidos	/orders	Lista para que los usuarios y los administradores visualicen los pedidos que tienen asociados. El superusuario visualizará todos los pedidos existentes en la aplicación.
Lista artículos	/articles	Lista para que los administradores visualicen los productos asociados a su comercio. El superusuario visualizará todos los productos existentes en la aplicación.
Añadir artículo	/articles/register	Formulario para añadir un nuevo artículo al comercio asociado al administrador que realiza esta acción.
Vista Artículos	/articles/{id}	Vista del artículo indicado en la URL
Modificar artículo	/articles/{id}/update	Formulario para modificar el artículo indicado en la URL.

**Tabla 11. Lista de rutas de la aplicación.**



## H Resultado interfaz gráfica de la aplicación

Este anexo recoge el diseño final de la interfaz gráfica realizada durante el desarrollo de este proyecto.

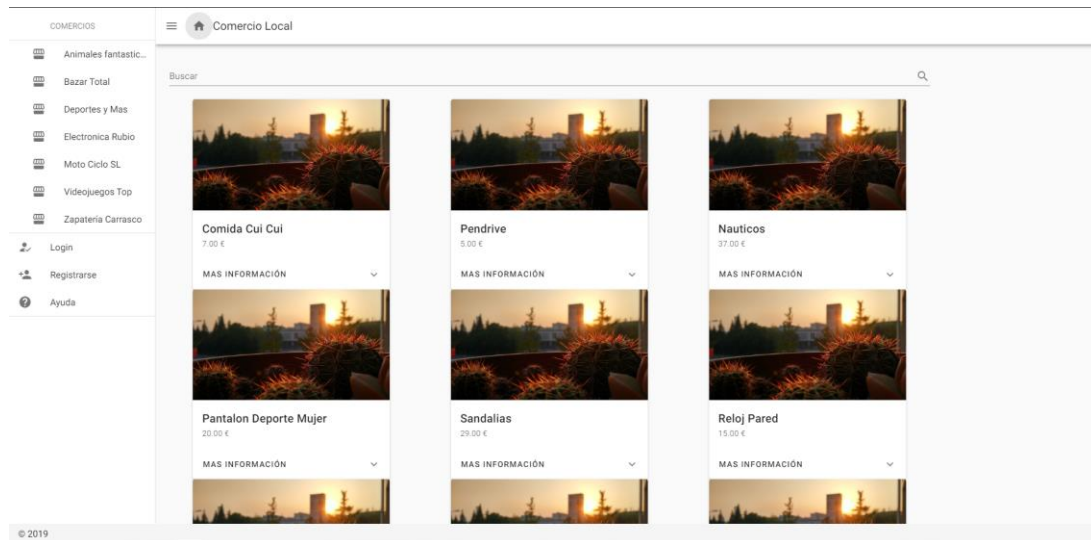


Figura 47. UI de la página principal.

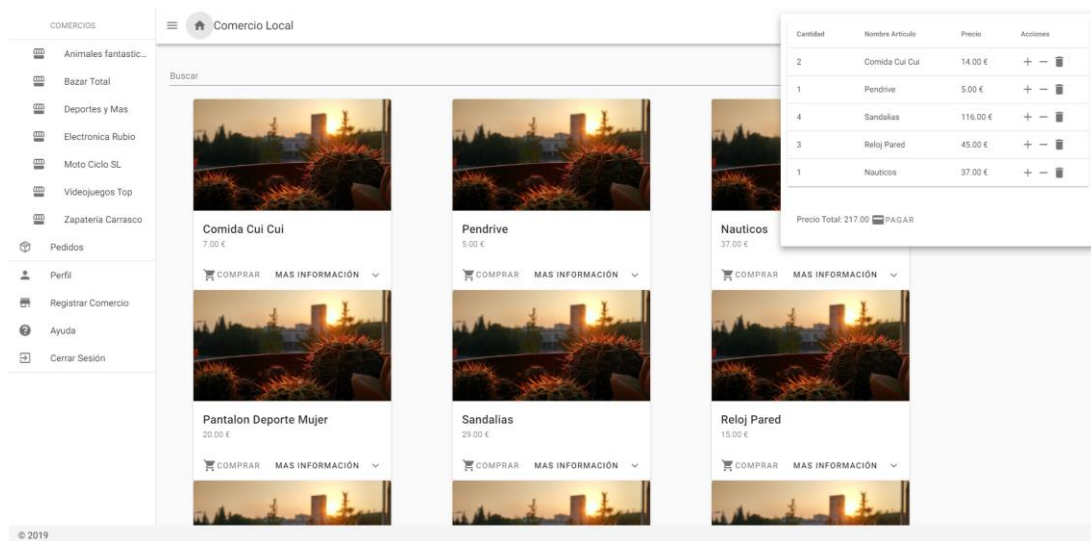


Figura 48. UI de la página principal con el carrito desplegado.

COMERCIOS

Animales fantastic...

Bazar Total

Deportes y Mas

Electronica Rubio

Moto Ciclo SL

Videojuegos Top

Zapateria Carrasco

Login

Registrarse

Ayuda

Comercio Local

Iniciar Sesión

Email

Contraseña

LOGIN

CANCELAR

© 2019

Figura 49. UI del inicio de sesión.

COMERCIOS

Animales fantastic...

Bazar Total

Deportes y Mas

Electronica Rubio

Moto Ciclo SL

Videojuegos Top

Zapateria Carrasco

Login

Registrarse

Ayuda

Comercio Local

Nuevo Usuario

Nombre

0 / 10

Apellido 1

0 / 10

Apellido 2

0 / 10

Email

Telefono

0 / 9

Contraseña

☐
 Aceptar condiciones para continuar con el registro

REGISTRAR

CANCELAR

© 2019

Figura 50. UI del registro.

COMERCIOS

Animales fantastic...

Bazar Total

Deportes y Mas

Electronica Rubio

Moto Ciclo SL

Videojuegos Top

Zapateria Carrasco

Login

Registrarse

Ayuda

Comercio Local

AYUDA

Ayuda 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris tempus libero et lectus tincidunt, et tincidunt tellus facilisis. Proin fermentum odio eu nisi auctor fermentum. Integer pellentesque efficitur sem, sit amet egestas massa iaculis non. Duis sollicitudin metus sed turpis rutrum cursus. Mauris in euismod purus. Vestibulum et nisl sapien. Vestibulum tincidunt elementum erat, nec euismod lorem ullamcorper in. Phasellus dapibus quis orci vitae semper. Proin eget tempor enim, et tempus tellus. Nulla a nulla quam. Praesent interdum eros eu nulla consequat, iaculis commodo quam placerat. Aliquam at facilisis dui, et tincidunt turpis. Morbi posuere odio sit amet mi aliquet condimentum. Nunc sodales diam sed ante efficitur, id vestibulum lorem tempor. Vestibulum luctus ipsum sapien, id mollis lacus varius in. Fusce mattis ex sem, gravida mattis nisl luctus at. Donec nisl augue, tempus eget enim ultrices, varius laoreet neque. Nunc nec porta erat, nec ornare quam. Sed egestas purus vel feugiat eleifend. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Suspendisse lacinia egestas ante, ac porttitor lectus consequat quis. Aliquam ac orci vel ligula gravida aliquam. Curabitur consectetur tellus a sapien ultricies, pharetra congue urna commodo. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Ayuda 2

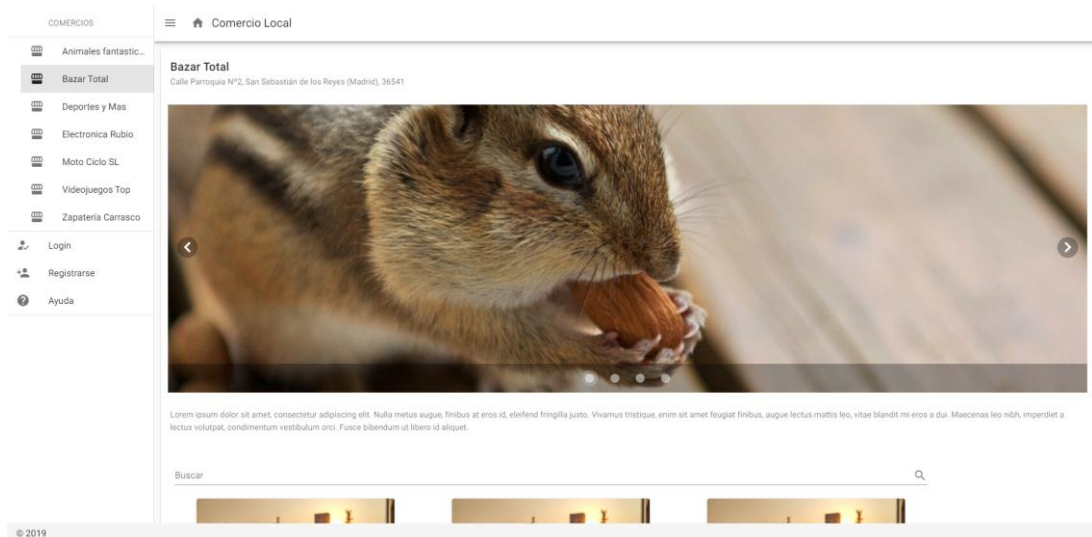
Sed quis mauris sem. Vestibulum mi massa, pulvinar id interdum nec, pulvinar id odio. Etiam velit turpis, hendrerit et ligula non, pellentesque fermentum nibh. Nunc sollicitudin rhoncus sollicitudin. Cras finibus mollis libero, et pharetra est suscipit molestie. Duis eget lobortis felis. Fusce malesuada volutpat diam, vel sagittis nulla molestie sed. Phasellus non ex vitae justo aliquam accumsan. Praesent facilisis gravida venenatis. Pellentesque vitae ligula ac felis finibus facilisis. Phasellus purus odio, ultricies nec posuere in, aliquam ac metus. Praesent bibendum nec dolor vel tincidunt. Praesent eget turpis a dolor imperdiet egestas. Etiam sollicitudin, mauris auctor elementum mattis, felis eros vulputate sem, nec luctus ligula risus quis metus. Praesent vel enim turpis. Etiam nec sodales ex.

Ayuda 3

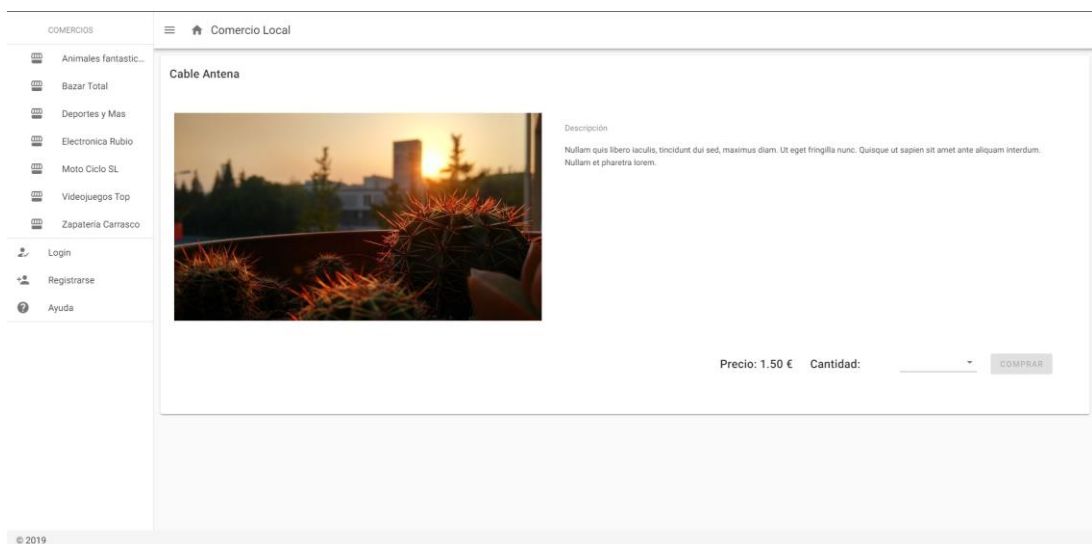
Aliquam erat volutpat. Curabitur vel enim eu velit tempus bibendum. Donec viverra vehicula magna, in laoreet risus placerat id. Cras a turpis nibh. Nullam dictum risus quis quam laoreet, eget vulputate elit porta. Sed pretium pellentesque nunc, tristique dapibus lorem consectetur id. Vestibulum eleifend arcu et neque dignissim rutrum. Cras vitae euismod urna. Quisque dapibus nisl quis ante feugiat fringilla. Nullam in libero in libero pulvinar iaculis. Cras vestibulum blandit ipsum id facilisis. Ut quam tellus, egestas eget vehicula quis, pulvinar sed velit. Curabitur auctor vulputate ex sit amet ullamcorper. Sed iaculis libero turpis, at varius mauris posuere quis. Curabitur fringilla quam sed fermentum mattis. Praesent ut ante felis. Proin sollicitudin sed purus ac dapibus. Nullam dignissim aliquet justo vitae vulputate. Mauris ligula odio, hendrerit id risus eget, bibendum ornare ante. Mauris sed nibh eget lectus venenatis vehicula.

© 2019

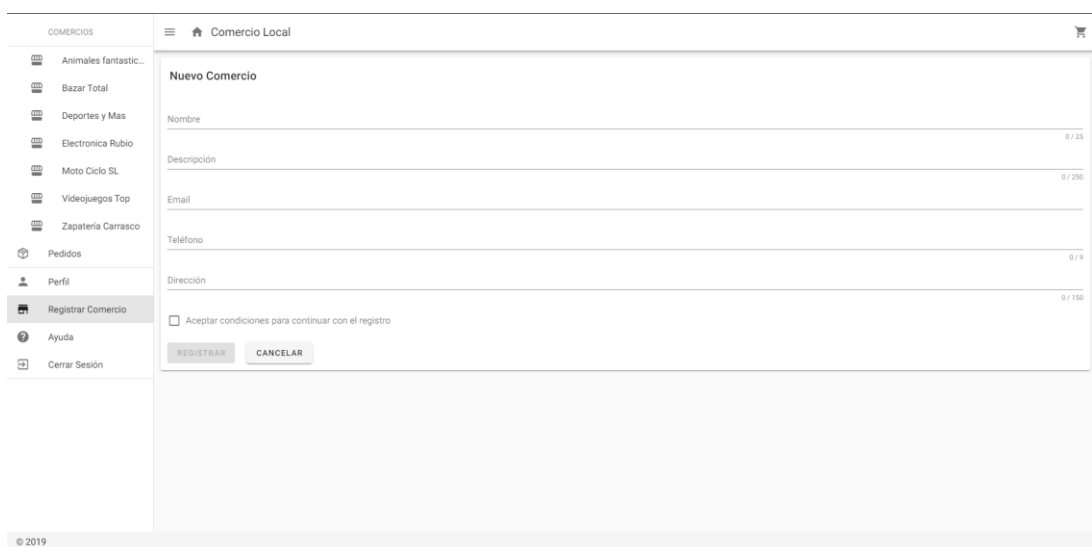
Figura 51. UI de la página de ayuda.



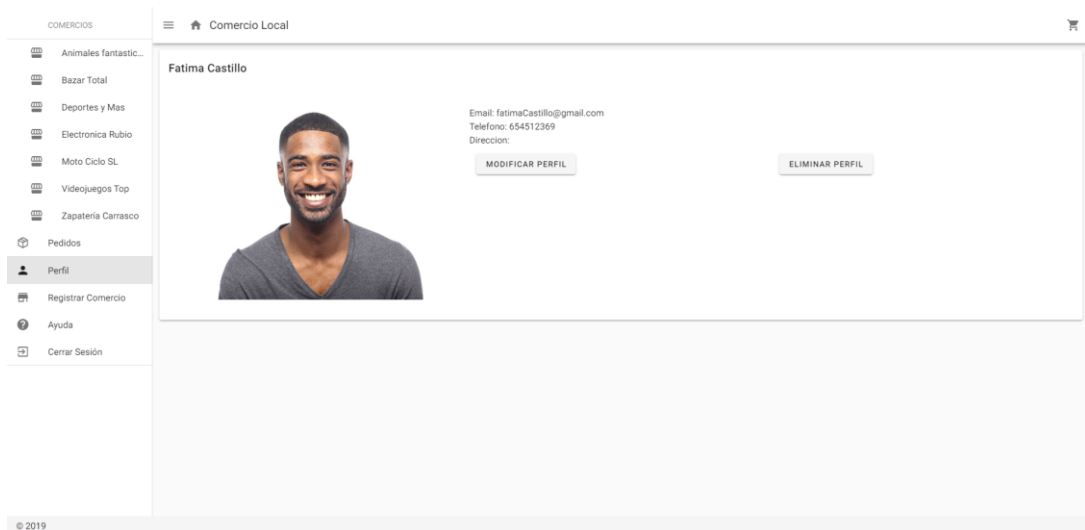
**Figura 52. UI de la vista de un comercio.**



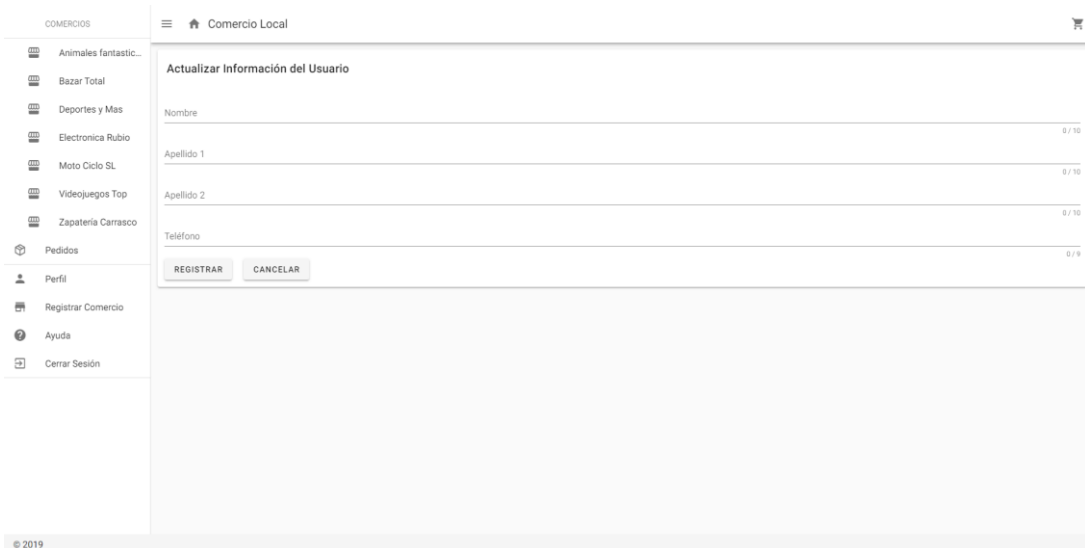
**Figura 53. UI de la vista de un artículo.**



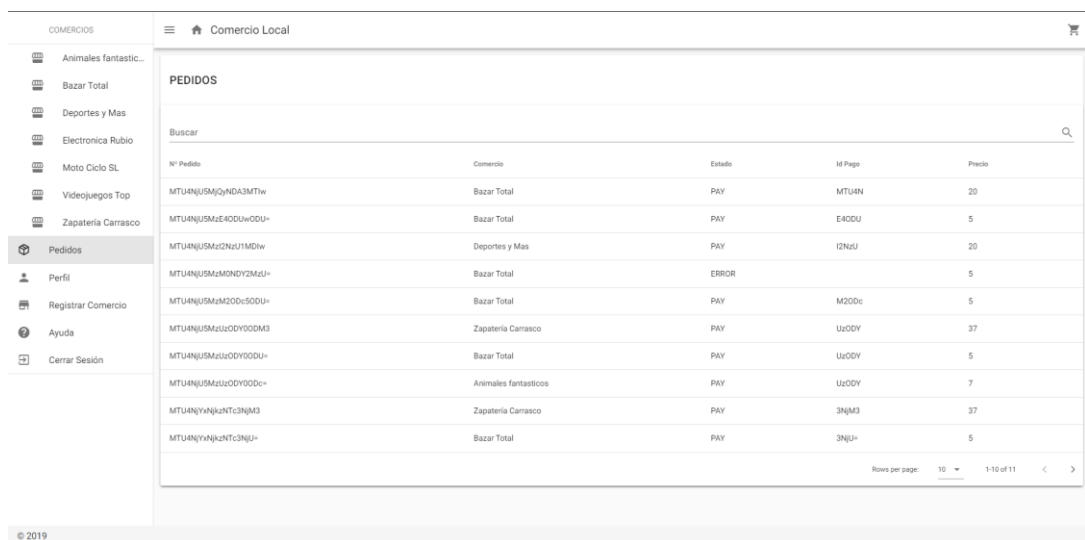
**Figura 54. UI del formulario de registro de comercio.**



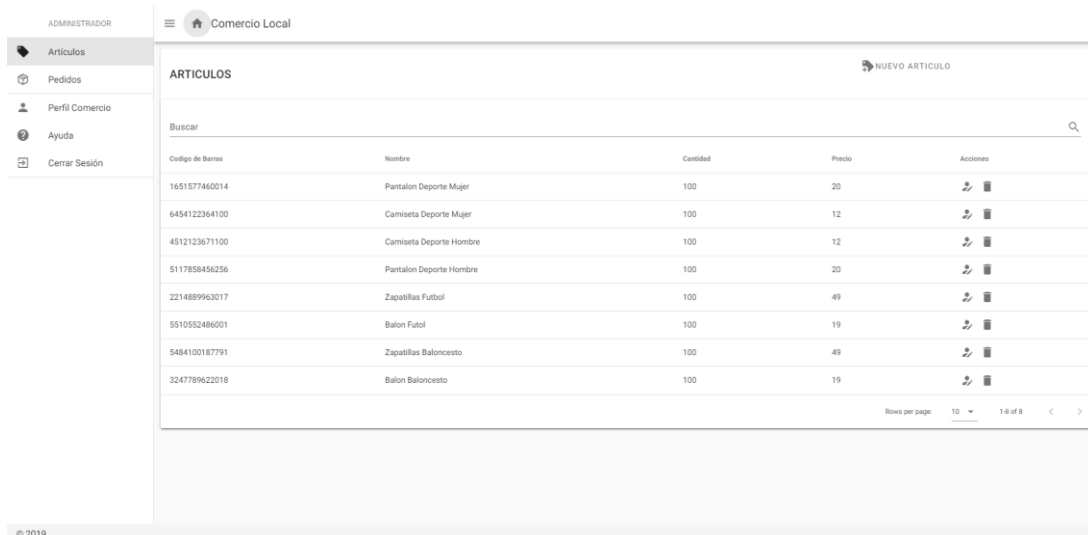
**Figura 55. UI del perfil de usuario.**



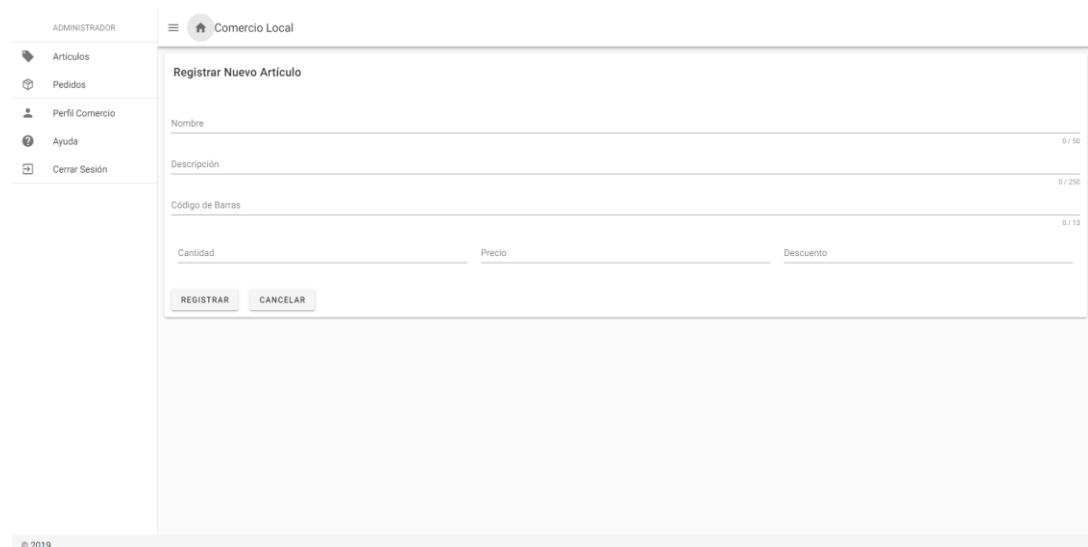
**Figura 56. UI del formulario de modificar un usuario.**



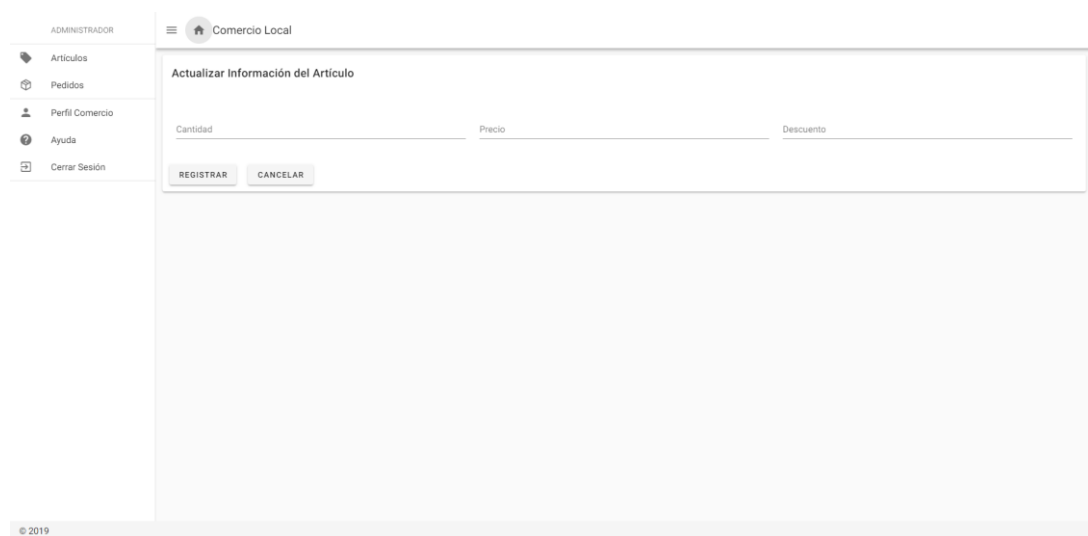
**Figura 57. UI del usuario de la lista de pedidos asociados.**



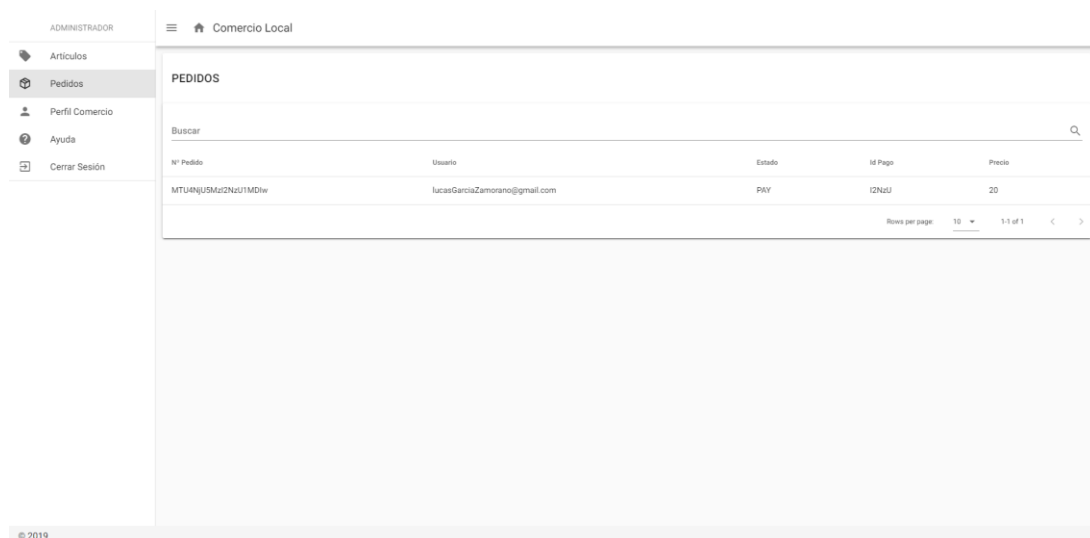
**Figura 58. UI del administrador de la lista de artículos asociados.**



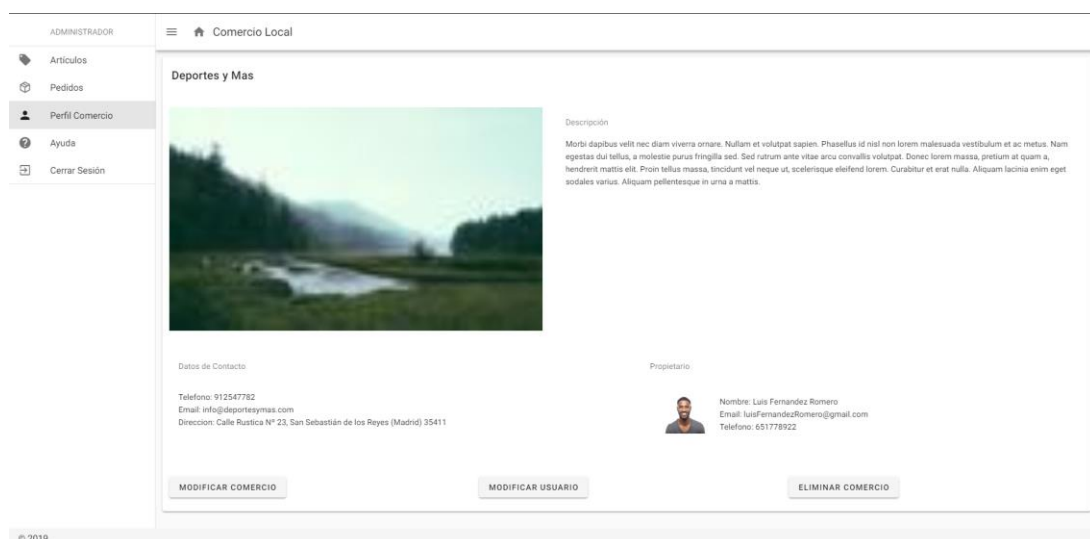
**Figura 59. UI del formulario para añadir un nuevo artículo.**



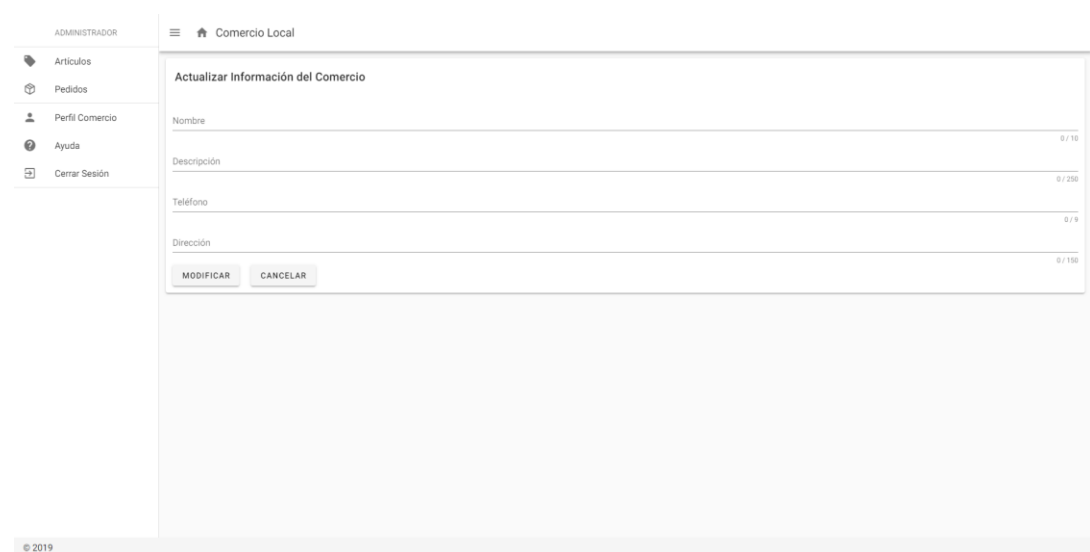
**Figura 60. UI del formulario para modificar un artículo.**



**Figura 61. UI del administrador de la lista de pedidos asociada.**



**Figura 62. UI del perfil de un comercio.**



**Figura 63. UI del formulario para modificar un comercio.**

SUPERADMINISTRADOR

Comercio Local

Artículos

Usuarios

Pedidos

Comercios

Perfil































Ayuda

Cerrar Sesión

USUARIOS

NUEVO USUARIO

Buscar

Nombre	Apellido 1	Apellido 2	Rol	Email	Telefono	Acciones
Lázaro	Augusto	Perez	admin	thelazaretor@hotmail.com	612345678	  
Manolo	Gomez	Prieto	admin	manoloGomezPrieto@gmail.com	651116841	  
Lucas	Garcia	Zamorano	user	lucasGarciaZamorano@gmail.com	645412697	  
davis2	2	2	user	davis2@davis.dav	986549954	  
Marta	Lopez	Carrasco	admin	martaLopezCarrasco@gmail.com	654785214	  
Luis	Fernandez	Romero	admin	luisFernandezRomero@gmail.com	651778922	  
Carlos	Rubio	Telido	admin	carlosRubioTelido@gmail.com	656844118	  
Alberto	Feliz		user	albertofeliz@gmail.com	654117478	  
Manolo	Masca	Ladrillos	user	manolete58@gmail.com	912345678	  
Fatima	Castillo		user	fatimaCastillo@gmail.com	654512369	  

Rows per page: 10 1-10 of 12

© 2019

Figura 64. UI del superadministrador de la lista de usuarios.

SUPERADMINISTRADOR

Comercio Local

Artículos

Usuarios

Pedidos

Comercios

Perfil

Ayuda

Cerrar Sesión

ARTICULOS

Buscar

Código de Barras	Nombre	Cantidad	Precio	Comercio	Acciones
5863445668753	Comida Cui Cai	99	7	Animales fantasticos	<div><div></div><div></div></div>
3657710028787	Pendrive	1	5	Bazar Total	<div><div></div><div></div></div>
1256451118696	Nauticos	99	37	Zapateria Carrasco	<div><div></div><div></div></div>
1651577460014	Pantalón Deporte Mujer	100	20	Deportes y Mas	<div><div></div><div></div></div>
6111377636889	Sandalias	100	29	Zapateria Carrasco	<div><div></div><div></div></div>
6211414550014	Reloj Pared	100	15	Bazar Total	<div><div></div><div></div></div>
3654112478222	Videjuego Fantasia	100	19	Videjuegos Top	<div><div></div><div></div></div>
2687410369681	Cargador Movil	100	5	Bazar Total	<div><div></div><div></div></div>
454446889520	Videjugo Guerras	100	19	Videjuegos Top	<div><div></div><div></div></div>
5456110548616	Zapatillas Deporte Niño	100	32	Zapateria Carrasco	<div><div></div><div></div></div>

Rows per page:

10

1-10 of 46

<

>

© 2019

Figura 65. UI del superadministrador de lista de artículos.

SUPERADMINISTRADOR

Artículos

Usuarios

Pedidos

Comercios

Perfil

Ayuda

Cerrar Sesión

Comercio Local

PEDIDOS

Buscar

Nº Pedido	Usuario	Comercio	Estado	ID Pago	Precio
MTU4NUSMzQNDzMTlw	lucasGarciaZamorano@gmail.com	Bazar Total	PAY	MTU4N	20
MTU4NUSMzE4ODUwODU=	lucasGarciaZamorano@gmail.com	Bazar Total	PAY	E4ODU	5
MTU4NUSMzI2NzU1MDIw	lucasGarciaZamorano@gmail.com	Deportes y Mas	PAY	I2NzU	20
MTU4NUSMzM0NDVlZmZu	lucasGarciaZamorano@gmail.com	Bazar Total	ERROR		5
MTU4NUSMzM2ODc5ODU=	lucasGarciaZamorano@gmail.com	Bazar Total	PAY	M2ODc	5
MTU4NUSMzUzODY5ODM3	lucasGarciaZamorano@gmail.com	Zapateria Carrasco	PAY	UzODY	37
MTU4NUSMzUzODY5ODU=	lucasGarciaZamorano@gmail.com	Bazar Total	PAY	UzODY	5
MTU4NUSMzUzODY5ODc=	lucasGarciaZamorano@gmail.com	Animales fantasticos	PAY	UzODY	7
MTU4NUSMzMTk4NzE3ODM3	davis@davis.dav	Zapateria Carrasco	PAY	MTU4N	37
MTU4NUSMzMTk5NTAzMjU=	davis@davis.dav	Bazar Total	PAY	MTU4N	5

Rows per page:

10

1-10 of 18

<

>

© 2019

Figura 66. UI superadministrador de la lista de pedidos.






















SUPERADMINISTRADOR

Comercio Local

COMERCIOS

+ NUEVO COMERCIO

Buscar

Nombre	Administrador	Email	Phone	Acciones
Animales fantasticos	davis@davis.dav	animales@animales.es	66666666	  
Bazar Total	manoloGomezPrieto@gmail.com	info@bazartotal.com	698412236	  
Deportes y Mas	kiaFernandezRomero@gmail.com	info@deportesymas.com	912547782	  
Electronica Rubio	carlosRubioTefido@gmail.com	info@electronirubio.com	998711003	  
Moto Ciclo SL	thelazaretor@hotmail.com	thelazaretor@hotmail.com	612345678	  
Videojuegos Top	monicaTonesPicado@gmail.com	gestion@videjuegostop.com	698412369	  
Zapateria Carrasco	martaLopezCarrasco@gmail.com	zapateriaCarrasco@gmail.com	654112298	  

Rows per page: 10 1-7 of 7

© 2019

Figura 67. UI superadministrador de la lista de comercios.